AD-A224 570

REPORT DOCUMENTATION PAGE

DTIC FILE COPY

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 86	Aug 1 83 - July 31 84

4. TITLE AND SUBTITLE

Center of Excellence in Aerospace Automation.

5. FUNDING NUMBERS

F49620-82-C-0092

(2)

6. AUTHOR(S)

Robert H. Cannon Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Stanford University
Stanford, California 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

AFOSR-TR- 90 0790

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Progress has been made on four components of our second generation intelligent system, SUCCESSOR:A) a geometric modeling system, B) an advanced symbolic graphics system, C) a system for matching structures from stereo paris and motion sequences, and d) ATLAS, a model-based planning system for automating the planning of assemblies development of common lisp were made to make our programs portable and to enable us to work on several computers. These developments indude: SLISP a common LISP subset, on VAX:b) TAIL, a version of SLISP on SUN workstations; c) a real time garbage collection algorithm for Lisp in support of robotics. In addition, small development of the AL programming system continued.

14. SUBJECT TERMS

15. NUMBER OF PAGES

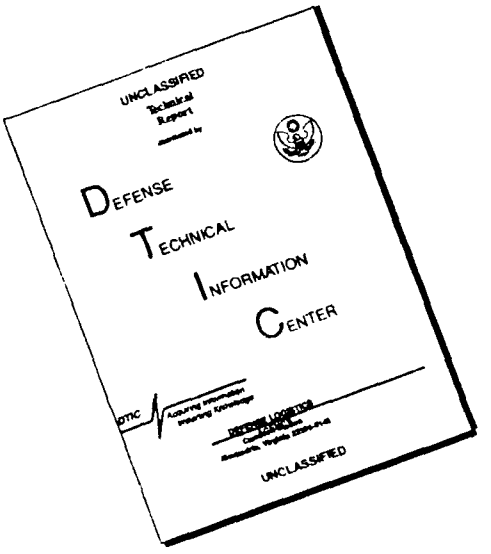
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified

NSN 7540-01-280-5500

Standard Form 298 (890104 Draft)
Prescribed by ANSI Std. Z39-18
298-01

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

ROBERT H. CANNON, JR.
CHARLES LEE POWELL PROFESSOR
AND CHAIRMAN
DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS

June 12, 1986

Major Joe Hager
Department of the Air Force
Air Force Office of Scientific Research
Bolling Air Force Base, DC 20332-6448

Subj: Interim Report on Contract F49620-82-C-0092

Dear Major Hager,

Enclosed please find a copy of the Second Annual Report for the Center of Automation and Manufacturing Science. This report, due on ~~September~~ 30, 1984, was completed in December of 1984, but because of an error in mailing was not received by your office. Please accept my apologies for this delay.

Sincerely,

R. Cannon (jg)

RHC:jg

cc: Anne Sprunt
Gerda DeWerk

ABSTRACT

The Air Force Office of Scientific Research has established, with major three-year funding, a Center of Excellence at Stanford University to develop new technologies that will be key to advancing automation of manufacturing processes. The new Center draws from two well-established research groups at Stanford: the Artificial Intelligence Laboratory's Robotics Group, and the Automatic Control Group in Aeronautics and Astronautics.

Our Air Force program focuses on the robotic aspect of automated manufacturing, which draws upon more of the new technologies, and more deeply, than any other aspect. We believe the next generation of lightweight, facile, quick, seeing, sensing, thinking, reasoning robots can provide the future flexible automation that will be so important in achieving higher levels of productivity. The needed underlying technologies fall into four categories: manipulator control, sensing, thinking, vision. We are working to make useful contributions in all four technical areas, with much interaction between, and synergism among us.

~~During this second year~~ Progress has been made on four components of our second generation intelligent system, SUCCESSOR: a) a geometric modeling system, b) an advanced symbolic graphics system, c) a system for matching structures from stereo pairs and motion sequences, and d) ATLAS, a model-based planning system for automating the planning of assemblies. Developments of Common Lisp were made to make our programs portable and to enable us to work on several computers. These developments include: a) SLISP, a common Lisp subset, on VAX; b) TAIL, a version of SLISP on SUN workstations; c) a real-time garbage collection algorithm for Lisp in support of robotics. In addition, small development of the AL programming system continued. ~~The AI Robotics Lab moved in May to larger quarters.~~

Development of control systems for our two-link manipulator with flexible tendons has progressed both in underlying theory and in hardware development. Full nonlinear (FORTRAN) simulation is operating, and advanced concepts for multi-input multi-output control are being tested. The two-link hardware has been tuned and calibrated. Initial control using end point feedback has been accomplished. Two extensive theoretical studies of task command strategies have been completed, and final reports published. Control system development for the unique configuration of a fast wrist on the end of a very flexible arm has progressed to where its utility in providing very-high-speed capability within a work station has been demonstrated in a series of pick and place tasks. We have begun basic work in one of the most difficult, and certainly one of the most important areas for deeply advancing robot manipulator capability: adaptive control.

This year's effort on integrated tactile sensors has concentrated on techniques for calibration of the tactile arrays that are being developed, with good implications for factory-robot calibration procedures that such techniques can facilitate.

Altogether, there are some 20 graduate student research assistants and fellows involved in this Air Force Research Program.

Table of Contents

INTRODUCTION

The Air Force Office of Scientific Research has established, with major three-year funding, a Center of Excellence at Stanford University to develop new technologies that will be key to advancing automation of manufacturing processes, with specific Air Force concern for assembly, test, and rework. These areas represent important economic leverage in the affordability of Air Force systems.

The new Center draws from two internationally known research groups at Stanford: the Robotics Group of Stanford's Artificial Intelligence Laboratory, and the Automatic Control Group of Stanford's Department of Aeronautics and Astronautics. A common objective that we are able to address together, with this major Air Force support, is to advance the effectiveness of automation in manufacturing by mounting research concurrently — and synergistically — into a set of the primary, pacing technologies in automation, as we outline below.

Our Center — the Center for Automation and Manufacturing Science (CAMS) — is the first of a new complex of centers at Stanford involved in the manufacturing enterprise: the Stanford Institute for Manufacturing and Automation (SIMA). The other founding Centers are CTRIMS, for graduate education in manufacturing operations, the Center for Design Research (CDR) which pursues creative use of the computer aided prototyping process and the Center for Metals Formability. We will interact in many ways with other centers at Stanford, such as the Center for Materials Research and the large Center for Integrated Systems of which Professor Meindl (a principal investigator on this AFOSR Center of Excellence program) is Codirector.

Within CAMS we are addressing a number of automation issues. For example, a major project for automation of ultra-high-precision machining is now underway.

In our Air Force program we have decided to focus on robotic aspects of automation. With their requirements for great flexibility of use and rapid task redirection, the robotic aspect of automated manufacturing will draw upon more of the new technologies, and more deeply, than any other aspect.

If the right set of technologies is developed, we believe the next generation of robots can (by comparison with today's) be lightweight, limber, deft, facile, quick, friendly, low-powered, seeing, sensing, thinking machines. Above all, they will be capable of reasoning and strategizing - of carrying out tasks assigned at a high conceptual level, by "thinking through" the best way to carry out any given task. Robotic devices with such characteristics and capability can provide the flexible automation that will be so important in achieving higher levels of productivity.

What are the underlying technologies that will be needed as the base for robots with such capabilities? They can be described in four categories: manipulator control, sensing, thinking, vision. Among us, in our Center, we are working to make useful contributions in all four technical areas. There is, of course, much interaction between, and synergism among the four areas; and that is the exciting thing about the level of effort that the AFOSR program makes possible. Specifically, fast, precise manipulator control is the primary focus of Task 3 of the program, tactile sensing of Task 4, and computer-based thinking

and vision of Task 2. But these depend upon each other altogether as diagrammed in Fig. 3-1, and draw upon one another in many ways. We feed back signals from many sensors — optical and eventually perhaps acoustic, as well as tactile and force — to effect good end-point control of manipulators. New, more competent manipulators, with their multiple sensors, will be utilized avidly by task-management systems to produce new assembly sequences that are quicker, more precise, and more efficient.

More acute robot vision, together with more rapid visual perception (scene analysis), are very important basics for more effective task planning, and may even someday be used in real time by the fast manipulator controllers themselves.

Intelligence and sensing are important throughout advanced manufacturing technologies. The Stanford Artificial Intelligence Lab (SAIL) has made progress in our work on inspection and machine vision, in manipulation, in mobile robots for warehousing, and in aspects of Computer Aided Manufacturing (CAM). AI can contribute to other areas of manufacturing, in design of mechanical parts, process planning, and cell control. Model-based systems with geometric reasoning contribute to all aspects of AI in manufacturing.

During the second year the Artificial Intelligence effort shifted from robot programming systems to model-based intelligent systems. Major components have been implemented of our second generation intelligent system, SUCCESSOR, following our first generation system, ACRONYM. These components are: a) A geometric modeling system was designed and built; it provides a user interface which simplifies construction of geometric models, capabilities which have proved to be important in recognising industrial parts; c) A vision system incorporates matching of structures for stereo pairs and motion sequences; d) ATLAS, a model-based planning system is aimed at automating the planning of assemblies; it was designed and partially implemented.

Lisp is the language of choice in AI. We have chosen Common Lisp as a basis for programs which are portable and which enable us to work on several computers. In support of implementation of intelligent systems, SLISP, a subset of Common Lisp was implemented on VAX in Franz Lisp. A version, TAIL, was implemented on low cost SUN workstations. The system may contribute to commercial implementation of Common Lisp. In addition, an algorithm was developed for real-time garbage collection in Lisp.

A small effort has brought a VAX version of the AL programming system for robots near completion.

The AI Robotics Lab moved in May to Cedar Hall. The move has given us adequate space for research. Altogether, the move was handled effectively; the effort and dislocation were reasonable.

Our advanced work on the rapid, precise control of robotic manipulators (refer again to Fig. 3-1) is being carried out in the Aerospace Robotics Laboratory of the Air Force Center at Stanford.

The underlying objective here is to develop the sequence of technologies that will enable future generations of robots to move much more quickly, more deftly, than today's robots, achieving much higher levels of precision, while at the same time removing the need for robots to be the heavy, rigid, power hungry machines that today's robots are.

Toward this objective, we are pursuing a sequence of specific projects, each with specific capability goals to be demonstrated, that will provide key elements of the desired new robot technology base. The AFOSR Center of Excellence level of funding has made it possible for us to make major advances this year, both in technological capabilities achieved and in the development of experimental facilities and expertise needed to carry on advanced research in this area.

During the second year of our Air Force funding, control system development for the unique configuration of a fast wrist on the end of a very flexible arm has progressed to where its utility in providing very high-speed capability within a work station has been demonstrated in a series of pick and place tasks.

We believe that the new capability for fast, precise control of the position of a gripper in space, despite its mounting at the end of a very flexible arm, represents a potentially very important advance for use in the next generation of robotic manipulators.

Development of control systems for our two-link manipulator with flexible tendons has progressed both in underlying theory and in hardware development. Full nonlinear (FORTRAN) simulation is operating, and advanced concepts for multi-input multi-output control are being tested. The two-link hardware has been upgraded, tuned, and the sensors calibrated. Initial control using end point feedback has been accomplished.

Two extensive theoretical studies of task command strategies have been completed, and final reports written. One is on the planning of minimum-time dynamic paths, and the other is on the use of feed-forward compensation to enhance speed and precision of target pursuit in factory assembly operations.

In keeping with our committment to use our AFOSR support to build the strongest possible fundamental technical base for robotics, we have begun basic work in one of the most difficult, and certainly of the most important areas for advancing robot manipulator capability: adaptive contol. This concept has been discussed and given theoretical attention for some years; but useful applications have been few (confined largely to chemical processing). By focusing our work on the specifics of robot dynamics, we intend to develop and demonstrate the contribution adaptive control techniques can make.

Finally, our special work on integrated tactile sensors (refer one more time to Fig. 3-1) has proceeded in the laboratories of Stanford's Center for Integrated Systems. This year's effort has concentrated on techniques for calibration of the tactile arrays that are being developed, with good implications for factory-robot calibration procedures that such techniques can facilitate.

Second-year progress of our work in each of these fundamental areas of robotic research is reported in some detail in the Technical Report Sections which follow.

Technical Report on Task 1

SURVEY OF KEY PROBLEMS AND TECHNOLOGY TRANSFER

This task has two parts, namely technology transfer, and setting up continued technical contact with industry personnel to bring industrial applications and requirements for consideration in research and design of systems.

Substantial progress has been made in both areas. SIMA, the Stanford Institute for Manufacturing and Automation, is now well-established. CAMS, established under this AFOSR support, is one of four centers in SIMA. Professor Cannon has been especially active in organizing SIMA. Through SIMA, working relations are being established with technical staff and management of a growing set of major corporations.

A major collaboration was carried out under the ITA program among Stanford, SRI, Adept Technology, and Honeywell Incorporated. The program was funded by DARPA and the Air Force, but a part was supported by this contract. The contract has had some very real technology transfer. Stanford force control technology was extensively transferred to Adept. Our older, more established force control algorithm, the Salisbury stiffness control, was taken to Adept in the form of an existing program. A PhD from SAIL worked in bringing it up on their own multiprocessor system. Extensive contact continues on force control, especially for our newer system, Khatib's Cartesian force control system. It is likely that technology transfer will continue for quite some time in the area of force control. A Stanford design for force sensing fingers which measure three components of force was taken by Adept which went on to modify the design to measure an additional component of torque to enable full six component force/torque measurement from a pair of fingers. They also modified the design for manufacturability. Also under ITA, one of our former students, Cregg Cowan, has gone to SRI where he brings them capability with ACRONYM and intelligent systems.

The Lisp developments supported by this contract have already begun to have some influence on commercial Lisp development. This is the best possible form of transfer, to a product, in this case Common Lisp. SLISP, the Common Lisp subset developed for our work, has been used by AI&DS, a local AI company.

A major disappointment was the failure of efforts to bring about a commercial version of AL, the programming system for robots. Even though versions of AL were built at Karlsruhe, Germany, Grenoble, France, and University of Tokyo, even though the version from Grenoble was commercialized and is used on the SEMI robot, even though Hitachi Production Engineering Research Lab has built a similar system, and even though AL has had a very strong effect on Japanese efforts for robot programming, there has been relatively little support available in the US.

Technical Report on Task 2

INTELLIGENT SYSTEMS FOR MANUFACTURING; INSPECTION AND VISION; SENSOR-BASED PROGRAMMING SYSTEMS

Overview

During the past year the AI effort shifted to focus almost entirely on model-based intelligent systems. In the past the Stanford Artificial Intelligence Lab (SAIL) developed the ACRONYM intelligent system. In the Intelligent Task Automation project (ITA) with support from DARPA and Air Force, we accomplished the following: 1) made major extensions to ACRONYM; 2) demonstrated success in modeling industrial parts, in predicting their appearance, and in identifying them; 3) determined requirements for future intelligent systems for inspection and vision.

Several sections below describe implementation of major components of our second generation intelligent system, SUCCESSOR. The first section describes an advanced symbolic graphics system which was designed and partially implemented. It will support very powerful geometric models. Several desired capabilities for ITA will be made possible by this system. Another section describes a system for constructing geometric models which was designed and built. It provides a user interface which largely meets the goals of learning to use the system in one hour and building models in fifteen minutes.

Another section describes ASTERIX, a subsystem which incorporates matching of structures of features for stereo pairs and motion sequences. It determines a strategy for ordering the matching process for clusters of features. A section describes ATLAS, a model-based planning system aimed at automating the planning of assemblies. A subset was designed and partially implemented.

A final section describes Lisp system support for intelligent systems. We have used SLISP in much of our work. It is a Common Lisp subset designed and implemented under this support.

A small effort has brought a VAX version of the AL programming system for robots near completion.

a. Intelligent Systems for Inspection and Vision

1. Graphics and Prediction from Models

Symbolic display of generalized cylinders

A symbolic display system has been designed and partially implemented. It generates a projection of visible surfaces of a very general subclass of generalized cylinders. The system is described in Appendix D. Conceptually it can be thought of as ray tracing, i.e. projecting back rays from each pixel to intersect objects, ordering intersections of surfaces along each ray by distance from the image. However, that can be improved on; it is wasteful to order surfaces along each ray, since order relations change only along boundaries, a one-dimensional subset of rays. In fact, depth relations change only at T vertices and cusps, a zero-dimensional subset of rays. This enables a striking decrease in computation in depth ordering of surfaces for hidden surface calculations.

Limbs of generalized cylinders are obtained by an iterative search stepwise along the surface. Surfaces are specified by spine, cross section, and sweeping rule, each of which may be an arbitary function of one parameter which is evaluated at each step. Steps are chosen according to a uniform quality criterion. The step distance was chosen to give a constant number of iterations per step. About 100 steps were required for a three turn helix. Objects are defined by unions, intersections, and set differences of these primitive volumes or surfaces.

Projections of surfaces are put into an image quad-tree from which surface ordering is obtained. T-junctions are found here.

The system has generated hidden surface views of complex parts, but the full system for structured objects is still being implemented.

From this research has come an approach to determining the locus of points at which the qualitative structure of projections changes. This may lead to compact prediction of image appearance for complex objects.

Building Geometric Models

Appendix E describes a system for building geometric models. The geometric modeling system has a user interface which relies on commands invoked from menus and selected with voice input. The keyboard is used for naming elements. Geometric forms are specified by points which are entered by pointing devices, in this case a trackball. Points are three dimensional points determined from stereo pairs of pictures. Generalized cylinders are specified by cross section, spine, and sweeping rule. Cross sections can be specified by a few points, e.g. three points for a circle or rectangle. For the class of generalized cylinders with straight spine and constant sweeping rule, only two cross sections are required. For some simple cross sections, a total of only four points are required for an object. For complex parts, still relatively few points are required, of order ten points. Parts can be defined from others by symmetry. A model of a simple object with a few parts can be built in 10 minutes. The system writes out a textual model of the part. It also determines object classes by generalization of constraints which determine object classes in ACRONYM.

2. ASTERIX: Stereo Matching and Optical Range Sensing

In Appendix F, Triendl describes a system for making correspondence between image structures in stereo pairs and motion sequences. Constraints in correspondence are maintained as separate knowledge sources which can change dynamically according to knowledge acquired in operation. Corners and curves are the features used in the system. It analyzes structures or constellations of these features in each image and plans a matching sequence which is chosen to be effective for the image. It classifies corners formed by curves, particularly to relax constraints on T-junctions which indicate occlusion and for which no correspondence is expected. The system groups features into similarity classes which might be ambiguous under the local matching operation. For example, for a checker board, interior corners are all similar, while corners at the four corners of the checker board are unique. The system begins by matching unique features. Corners connected to them are then unique. Matching takes place between classes of features, rather than individual features. Matching has been tested on several images and on artificial data.

Appendix G describes the analysis of ranging methods to derive a quantitative comparison of several methods for measuring range: a) structured light; b) triangulation using a laser diode and position sensitive detector; c) time of flight by phase measurement of a modulated laser source.

Appendix H describes a method for approximate Gaussian convolution by an algorithm which is suitable for implementation in simple, high speed hardware.

b. Intelligent Programming Systems for Robots and Manufacturing: Experiments with Implemention of Atlas

In [Brooks and Lozano-Perez 83] we proposed a new approach to task-level robot programming. Task-level programming attempts to simplify the robot programming process by requiring that the user specify only goals for the physical relationships among objects, rather than the motions of the robot needed to achieve those goals. A task-level specification is meant to be completely robot-independent; no positions or paths that depend on the robot geometry or kinematics are specified by the user. Appendix I describes experimental implementation of a part of ATLAS.

We implemented parts of the ATLAS system (designed in the above referenced paper); Automatic Task Level Assembly Synthesizer. There were two aspects to this implementation:

- Implementation of planning submodules.
- Integration of the submodules.

We took two target assemblies; bolting a lid on a box, and inserting and bolting an internal bracket on a floppy disk drive

The planning submodules we used were

- A CAD-like modelling system based on prisms, which included tolerance information.
- A 2 finger grasp planner (based on [Laugier 81]).
- A symbolic constraint system (based on [Brooks 81]).
- A goal state specification module.

These modules were partially integrated by using a plan skeleton matcher. A series of intermediate goals specified by the user was automatically turned into a sequence of partially specified plans to be fed to the above specialized planners. This integration highlighted problems with using complex algebraic constraints as a communiction mechanism.

We have not yet achieved full synthesis of our target plans.

The results of the integration attempt have led us to consider new bounding primitives for a geometric based constraint system.

c. Lisp System Support for AI

In developing major software systems, it is important to plan for computer portability. The hardware available will change rapidly and may have revolutionary changes over the life of the system. We have chosen to develop software in a subset of Common Lisp.

Lisp system development has been directed by Prof. Brooks. This section describes: I. implementation of a Common Lisp subset; and II.

I. Common Lisp subset

A subset of Common Lisp, SLISP, was defined and implemented in Franz Lisp on VAX systems. A version, called TAIL, was implemented for SUN workstations under the V kernel.

Common Lisp [Steele 84] has been adopted by DARPA as the standard language for Artificial Intelligence. There are a few implementations of Common Lisp underway for various machines, but in 1983 there were none for small workstations such as the SUN. (This situation has radically changed as a result of our work. At least three companies that we know of, namely Frans Inc., Lucid Inc., and Gold Hill Computer have been inspired by our work to seriously pursue Common Lisp on 68000 based machines.) The only other Lisp for a SUN-class workstation is Franz Lisp. Frans has a simple compiler, an inefficient function call mechanism, and poor performance on numeric computations.

TAIL is a very efficient Lisp with an efficiency profile tailored towards robotics applications.

The implementation is almost complete. We have a Lisp with:

- a novel, highly optimizing compiler.
- an assembler.
- a Common Lisp stream-based I/O system.
- an interpreter.
- a subset of Common Lisp functions.
- a garbage collector (copying compacting).
- an integrated interface to V kernel windows.

The TAIL system provides a useable and efficient Common Lisp subset for Stanford researchers to base their systems upon.

Besides the utility aspect of the project there were two research aspects of the TAIL project:

1. Compiler technology:

It was intended that TAIL would be transportable to other, and new, architectures so that a uniform environment could be provided for robotics research in the face of ever changing hardware.

This intent provided the driving forces for the following developments:

a. The source to source analysis and optimization of Lisp programs developed by [Steele 1978?] and extended by [Brooks, Gabriel and Steele 1982] was further refined. In the TAIL compiler there are two source to source phases followed by a code generation phase. The phases are:

- Alphatize.
- Analyze.
- Generate.

1. Alphatization carries out macro expansion, implicit PROGN re-introduction, constant folding (in a general manner), special binding analysis, and simple code re-ordering.
2. Analysis traces all variable, block and tag references, annotates all data flow within functions, carries out detailed side effects analysis, and does register usage analysis. It produces a summary of this analysis for the code generator. In addition it both introduces and eliminates variables, removes value discarding side-effectless code, re-orders evaluation where possible to aid register allocation, and occassionally re-invokes phase 1 when its source to source transformations have been large.
3. Generation is machine independent: see below.

b. Register allocation is done in a machine independent way, based on a register class description for a particular machine and the register annotations in code templates (see below). This is carried out as part of side effects analysis. When too many registers are required, a source to source transformation of the Lisp code is done in a manner which guarantees fewer registers will be needed.

c. A machine independent code generation strategy was developed. It differs from earlier strategies in its combination and extension of approaches. It defines a virtual machine for control aspects of Lisp (e.g. function call, returning multiple values, etc.) which is implemented as LAP macros. It uses code templates, annotated with memory and register class requirements, to generate code for data manipulation functions (e.g. car, svref, string-length etc.). The result is that 83% of the compiler source code is independent of the target machine.

2. Foreign Window Interface.

TAIL relies on the V-kernel window system for its user interface. This is in contrast to the approach taken by Lisp Machine manufactures who build a new window system within their Lisp, claiming that it is necessary to get a tightly coupled interface. Our experience shows otherwise. Our approach required integration of foreign language function call, and extensions to Common Lisp stream primitives to handle windows. At the Monterey Common Lisp meeting in September 1984, DARPA set up subcommmittees to define standards for foreign function call and window system in Common Lisp.

II. Lisp techniques for real-time control.

A traditional Lisp system with a stop-and-collect garbage collector can only guarantee that CONS will take no longer than a worst case complete garbage collection. This can be unacceptable for a program which has even moderate real-time response requirements and only a small heap to collect (e.g. a robot supervisor, an applications signal processor, or a message switching node). It can also be unacceptable in an interactive mode especially on a machine with a large virtual address space where, although infrequent, garbage collections can effectively shut the machine down for many minutes.

A real-time garbage collector runs concurrently with user Lisp code collecting garbage so that there is always free storage available to the user program — effectively providing a relatively small bound on the time necessary for CONS.

There are two major classes of proposals for real-time garbage collectors.

One class uses parallel processors [Steele 75] with at least one for the garbage collection process, and one for the user process. They make use of special hardware to provide synchronization on access to pointers. The garbage collection algorithm consists of three phases: mark, sweep, and relocate. If the two processors were to be simulated on a single processor the synchronization primitives would be unnecessary, but there is still an extremely large overhead on stock hardware in having the standard Lisp primitives (e.g. CAR, CDR, RPLACA, RPLACD, and EQ) check on the phase of the garbage collector. We are unaware of any such garbage collectors (parallel or serial) having been built.

The second class [Baker 78] is more inherently serial. Lisp primitives occasionally call on the garbage collector to run for a bounded amount of time. The collector has only two phases: sweeping which is done in chunks at the invitation of the user process, via Lisp primitives, and flipping which takes a bounded amount of time, during one of the sweep chunks. Such algorithms have been implemented, but they all use special purpose hardware and micro-coded primitives to achieve moderate performance. The performance costs of implementing them on stock hardware seems extremely high. "Stock hardware" refers to common modern architectures for Von Neumann uni-processors (e.g. MC68000, IBM370, VAX, NS32032, etc.).

We analyzed the above problems and [Brooks 84] proposed a new storage layout for all heap allocated Lisp/ objects which makes implementation of a Baker-style real-time garbage collector look much more attractive on stock hardware. It looks attractive enough to be practical in a wide range of cases, discussed below in section 3. The storage scheme might have beneficial effects for micro-coded machines also.

We first showed that copying compacting real-time garbage collection algorithms do not always need to protect user programs from seeing uncopied data, so long as a slightly more complicated collection termination condition is used. This opens the way to adding an indirection pointer to all Lisp heap objects making it unnecessary to check the garbage collection status of pointers, and so the overhead costs for most primitives reduce to the addition of a single instruction. Impure primitives, i.e. those which write into existing structures (e.g. RPLACD), have their overhead reduced by a factor of almost 2. Code density is correspondingly decreased. The cost of this scheme is increased storage size for all Lisp heap objects—one extra pointer per object, although for some objects (e.g. floating point numbers) this expense is already necessary in many other non real-time garbage collection algorithms.

We carried out a quantitive analysis [Brooks 1984] of the effects on code density and primitive runtime for a 68000 based Lisp. We did not carry out any implementation experiments. Such experiments would be necessary to fully validate the approach and its constant factor real-time cost.

Technical Update on Task 3

RAPID, PRECISE CONTROL OF NONRIGID MANIPULATORS

Overview

The underlying objective here is to develop the sequence of technologies that will enable future generations of robots to move much more quickly, more deftly, than today's robots, achieving much higher levels of precision, while at the same time removing the need for robots to be the heavy, rigid, power hungry machines that today's robots are.

Toward this objective, we are pursuing a sequence of specific projects, each with specific capability goals to be demonstrated, that will provide key elements of the desired new robot technology base. The AFOSR Center of Excellence level of funding has made it possible for us to make major advances this year, both in technological capabilities achieved and in the development of experimental facilities and expertise needed to carry on advanced research in this area.

The flow and interaction of our projects is indicated by Fig. 3-1 (which is repeated for convenience from Reference 2), where the AFOSR-funded portion of our research is shaded blue. Our most basic work is done under our AFOSR support; in a real sense our other work – – on robot task-related technology for DARPA and on manipulators in space for NASA, – – as well as for industrial robot designers, derives from the series of basic capabilities we are able to develop with our AFOSR support.

In that sense, our primary current experimental facilities are the two-link arm with flexible tendons, with which we are developing the capability for fast, precise control in two dimensions using end point sensing, and the very flexible arm with a quick wrist at its end, with which we are advancing the ability for very quick local task performance. This year's progress in these two central areas is described below. It will be seen that the program of research with the two-link arm is about on the (approximate) schedule of Fig. 3-1, while the quick-wrist research is about one-half year ahead of schedule.

We have also completed, this year, two theoretical and simulation studies into robot manipulation strategies ("moving-object catch strategy" in Fig. 3-1), which culminated in Ph.D. theses as reported below.

We have started, this year, upon the long-term and quite pervasive program in adaptive control of robot manipulators that is indicated beginning at the top center of Fig. 3-1. This will provide an important new technology base for much of our specific future research projects, as Fig. 3-1 indicates.

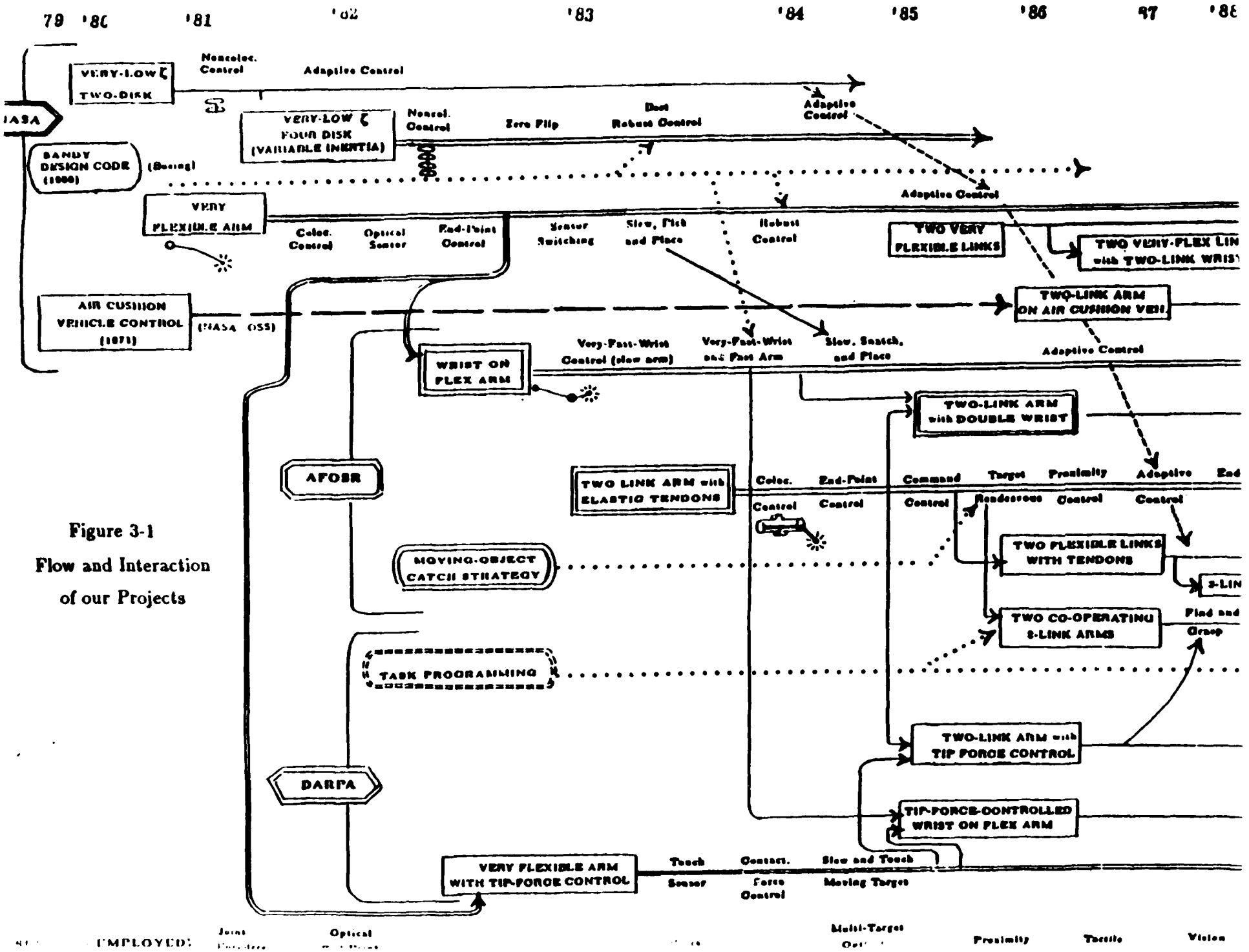


Figure 3-1
Flow and Interaction
of our Projects

a. Fast Wrist on a Flexible Arm

This section continues Section 3(a) of Ref. 2, pp 32-37, on the research led by Wen-Wei Chiang.

New Progress

In addition to the fast controller for the wrist motion described in Ref. 2, fast controllers for the flexible arm have now also been designed and tested. The fast controllers for the flexible arm use a state estimator and state feedback to achieve high performance.

The wrist-beam system was tested with its wrist loop closed in order to determine the system dynamics when the hub motor is used to control the motion of the flexible arm. Two different system conditions were tested, and a total of three controllers have been designed for them.

Here, the first system condition is that the tip is within the field of view of the optical sensor and near the target position, the wrist motor loop uses the wrist-tip position sensor for feedback control, and the wrist-axis RVDT is used for the state estimator of the flexible-beam controller. One controller was designed for this system condition.

The second system condition is that the tip is far from the target position. In this case, the wrist is regulated to a fixed wrist angle using the wrist-axis RVDT sensor. The wrist-tip may be either inside or outside the field of view of the photo sensor in this condition. The wrist-tip-position sensor signal can be used for the control of the flexible arm when inside the field of view; otherwise, the hub potentiometer of the flexible arm will be required. Two different controllers were designed for second system condition, one uses the wrist-tip position sensor and the other one uses the hub potentiometer.

Derived rate information obtained from the hub potentiometer is also used in all three different controller designs. Using the hub rate information for the control of the flexible beam has the advantage of higher loop gain, as indicated in the earlier study for the control of a one-link flexible arm.

An electronic circuit was built recently to process the wrist-tip-position sensor signal, and obtain its rate information for better control. Larger damping was achieved in the wrist-tip position control, and overshoots were eliminated since the more powerful position-plus-rate feedback could be used in place of the lead compensation on the position feedback alone.

A numerical analysis has been conducted to gain more understanding of this behavior of such a wrist-beam system. The rigid wrist was assumed very heavy, about half the weight of the flexible beam, so that the interaction between the wrist-tip motion and the structural flexibility became larger. This assumption made the analysis more general because a large actuator may be required for a mini-manipulator to handle heavy loads. This numerical analysis allowed a theoretical study about the ultimate performance of a control system, because perfect linearity and perfect knowledge about the system dynamics could be assumed.

During the numerical analysis, controllers were designed using different methods, and

their performances were compared. It was found that the method used in the experiment, by first closing the wrist motor loop with the end-point sensor, behaves as well as the other methods such as LQG and pole-placement technique; but closing the wrist loop first is believed to be most robust (less sensitive to parameter uncertainty).

New Demonstrations

In this second year, three new demonstrations were performed using the new improvements and designs, and their motions are recorded in the Figures 3-2, 3-3, and 3-4:

1. Figure 3-2 shows experimental fast motion of the wrist tip between three nearby target positions, while the flexible arm is controlled by a fast controller to align itself with the wrist. In the experiments, a gripper on the end of the wrist is carrying out (with high reliability) a "shell-game" type of pick-and-place demonstration, which can be viewed in an available videotape.
2. Figure 3-3 shows large slew motion of the system from a position outside the field of view of the tip-sensor to a position inside the field of view. The flexible arm's controller switches from one algorithm to another, depending upon the sensor configuration. The smoothness of the motion between switching sensors and/or algorithms depends upon the accuracy of the system model dynamics, the sensor linearity, and the signal continuity between different sensors in their overlapping boundary. For this demonstration calibration, against the hub potentiometer, of the tip-position photo sensor at the edge of the field of view was used to ensure smooth transition at that edge.
3. Figure 3-4 shows target snatch by the wrist when the large flexible arm is moving through the target area without stopping. The fast wrist stops its tip in space at the target position for *0.2 sec.* to allow the gripper to "snatch" the target object, but the arm keeps on moving. This is another very important capability. As humans we do this frequently; but, so far as we know, this the first demonstration of a robot manipulator doing it, and rather smoothly!

Some ripples at *10 Hz* showed up in the demonstrations. The *10 Hz* motion is the third structural mode coupled with the vertical bending and twist mode. It is neither within the control bandwidth of the wrist, which was designed for *4.5 Hz,* nor was it modeled in the controller design for the flexible arm. The demonstrations showed that the wrist-tip positional response behaved well within its designed bandwidth.

Referring again to Ref. 2, p. 36, we have now accomplished tasks (1) and (2) (first year) and (3), (5), (6), and (7) (second year) of the goals laid out for this part of our research. The remaining two tasks, (4) and (8), and other plans for the next year are described as following:

Future Plans

Task (4), "Target tracking and tip velocity control" is planned to be performed in three parts. First, position and velocity commands for the wrist tip are generated by the control computer or by an external signal generator, and the system is controlled to follow the command signals. This test will verify the bandwidth and accuracy of the control

system. No hardware modification is required, and this part should be done first.

Second, the static target-position identification. The positions of non-moving targets are indicated by the same kind of incandescent lamp mounted at the wrist-tip. The lamps will be turned on one at a time, and the same photo sensor is used to measure their position, so that the wrist can be brought to the right place to pick up targets. We have started building the equipment required for this task.

Third, the dynamic target-position identification. The wrist can track a moving target if their positions are measured simultaneously. This task requires replacement of the incandescent lamp with two sets of LEDs, one for the wrist and one for the target. The LEDs are modulated at different frequencies so that they can be detected by the same photo sensor simultaneously. The LED system will be similar to that implemented in the two-link arm system, and it is also planned for mounting on the new wrist-arm system for force control.

Task (8), "Contact Force Control" is planned to be tested on a new wrist-beam model. The new model incorporates several improvements, including the selection of a larger wrist motor, the addition of a force sensor, and modulated LEDs. Force control of a flexible arm was originally performed under DARPA contract, and the study of using the wrist to improve the performance in force control will also be partially supported by the DARPA contract.

We believe that the new capability for fast, precise control of the position of a gripper in space, despite its mounting at the end of a very flexible arm, represents a potentially very important advance for use in the next generation of robotic manipulators. The study of this project demonstrates how a mini-manipulator, such as a dextrous hand with multiple fingers, can be operated at a bandwidth higher than the limitation set by the structural flexibility.

Again, to see this potential most clearly in the industrial robot context, consider that a typical industrial robot arm might be 100 times stiffer than our demonstration one, and accordingly divide all the time scales in Figures 3-2, 3-3, and 3-4 by ten!

A study will be performed to investigate the feasibility of making a two-dimensional wrist/mini-manipulator, and add it to the two-link arm for improved position and force control in two dimensional space. Accurate position and force control are essential for the project of two cooperating arms, which has been proposed to DARPA recently.

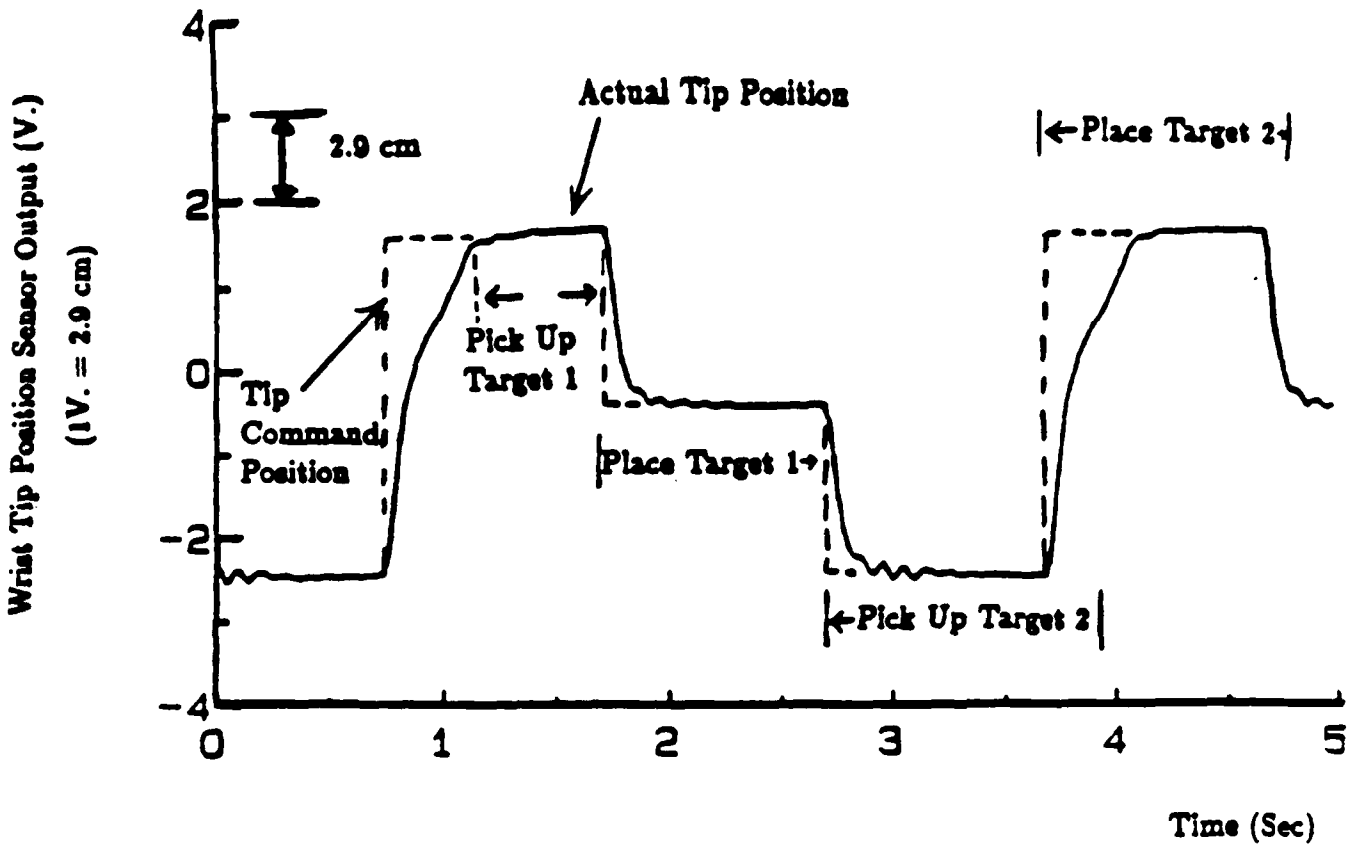
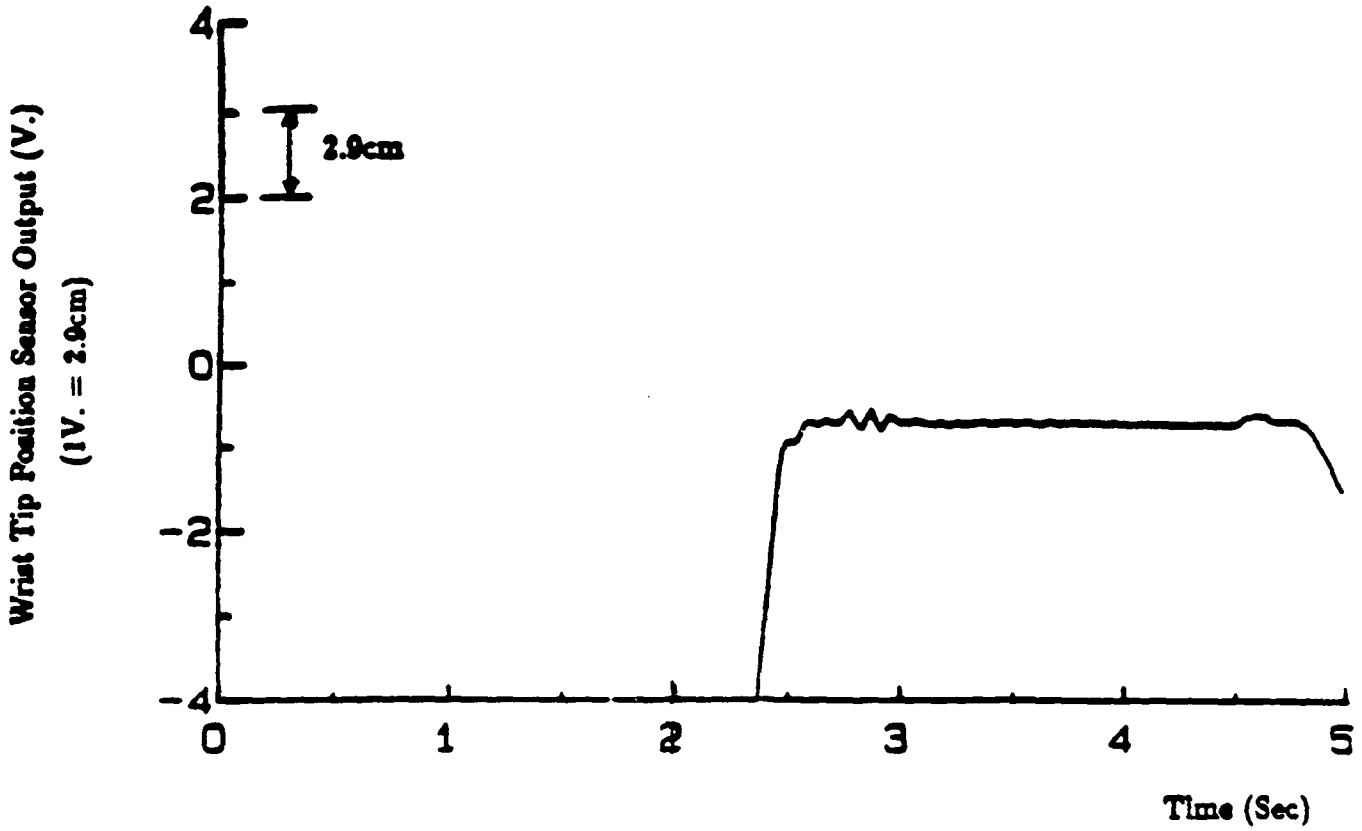


Figure 3-2 Fast motion between three target locations

"Shell-Game"



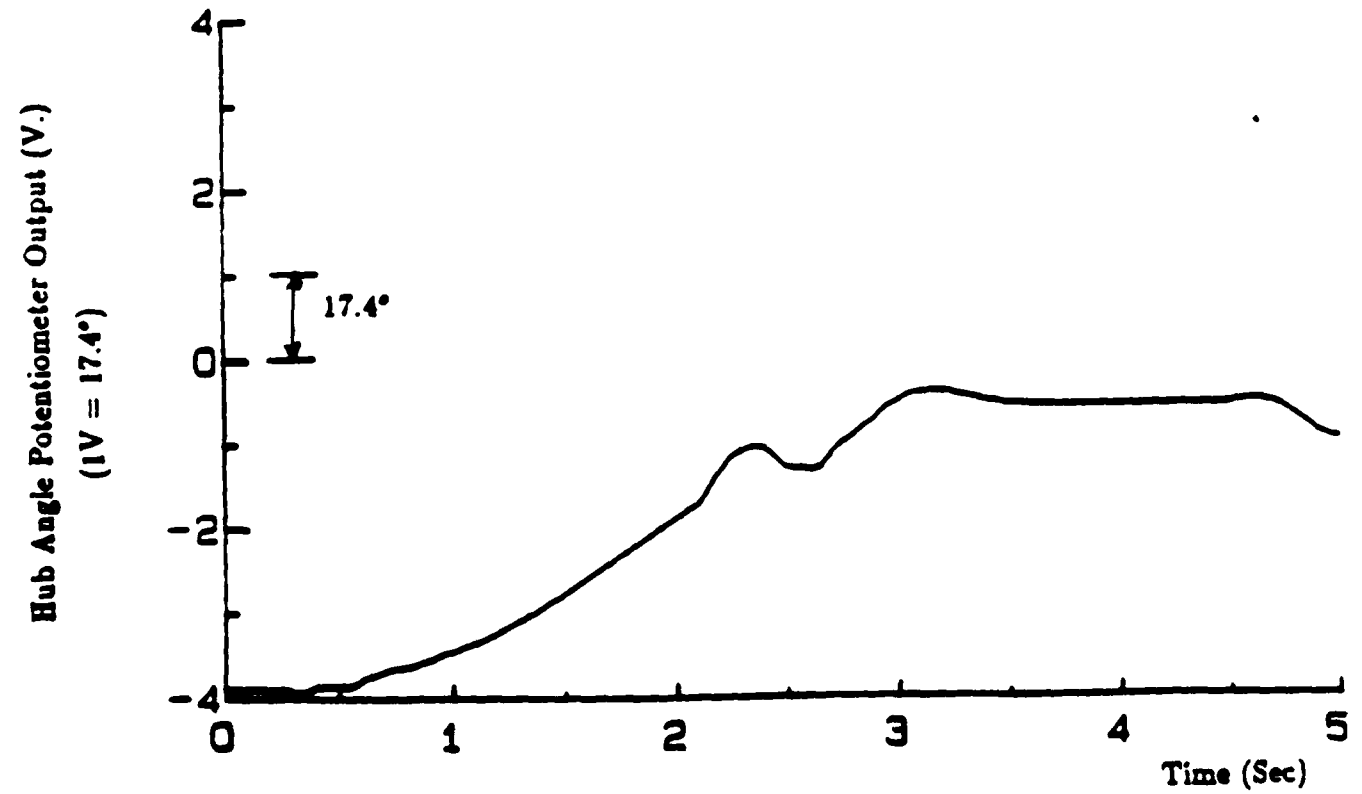


Figure 3-3 Large slew motion

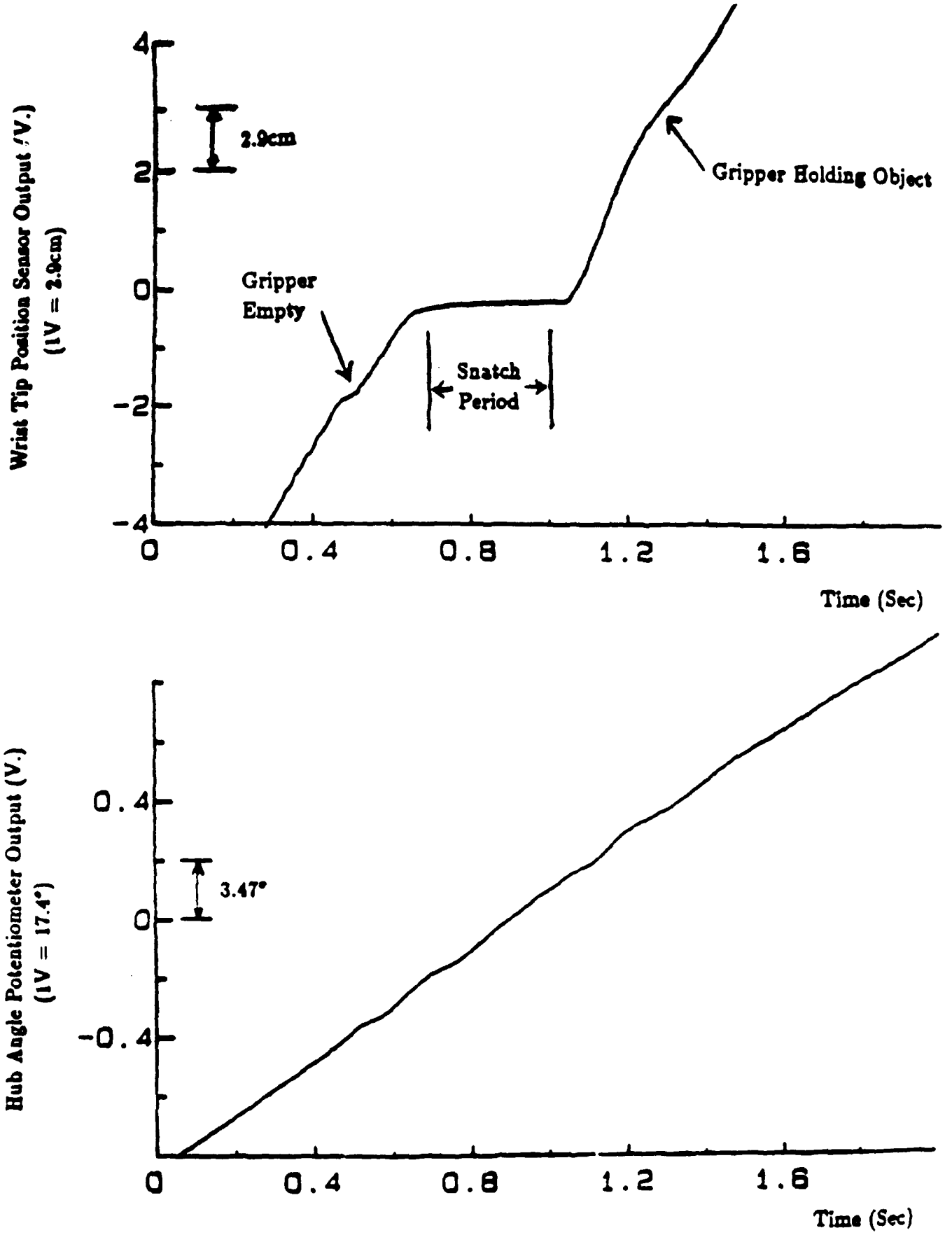


Figure 3-4 Target Snatch

b. Two-Link Manipulator with Flexible Tendons

This section continues Section 3(b) of Reference 2, pp. 37-40, on research led by Michael Hollars and Larry Pfeffer.

Theoretical Analysis

The two-link arm represents a significant increase in theoretical complexity over the one-link flexible arm. The complexities are introduced in two main areas. First, the two-link arm dynamics or equations of motion are highly non-linear and can NOT be trivially linearized as for the one-link arm case. Second, the two-link arm system is inherently multi-input, multi-output (MIMO), since the dynamics are highly coupled and cannot be decoupled into smaller single-input, multi-output (SIMO) systems.

There is one area where the first experimental two-link arm has intentionally been made simpler than the one-link experimental arm: The system flexibility has been discretely lumped in the form of springs in the drive train instead of continuously distributed in the structure. Thus, the problem has been initially simplified in one respect so that we can more easily absorb the new complexities of two links. Later versions of the two-link arm will include continuously flexible structural elements.

The next two paragraphs outline the theoretical considerations behind the three complexities introduced above.

The two-link arm dynamics are nonlinear in three different ways: First, the inertia of the arm as "seen" from the shoulder joint is a cosine function of the elbow joint angle. Thus, pulling the arm in will decrease the overall arm inertia and system vibration frequencies will increase (since the spring constants don't change). Second, there are Coriolis and centripetal accelerations present which are functions of the square of the link angular velocities. These nonlinear acceleration terms are roughly the same order of magnitude as the linear acceleration terms, and cannot be ignored. The third, major, nonlinearity is the large variation in inertia due to large changes in payload mass.

These nonlinear dynamics have always been present on robotic systems built in the past, but could be accommodated in an approximate way with little adverse impact on performance. The unique features of our manipulator, which require meticulous consideration of these nonlinearities, are the presence of variable flexible modes, the large payload-to-arm-mass ratio (1:1 for our arm compared to 1:10 for most arms), and especially the use of end-point sensing. The full nonlinear equations of motion of the two-link arm with flexible tendons is derived and written as FORTRAN simulation code in Appendix J.

The multi-input, multi-output (MIMO) aspect of the two-link arm (along with the quick-wrist system, Section 3a) represents a major step for our robotics lab. Most of our previous experiments with one-link manipulator position and touch control have been single-input, multi-output (SIMO) systems which can be treated with classical control analysis and design techniques (such as Root Locus, Bode and Nyquist). There are no similar techniques for MIMO systems. State space methodologies such as Linear Quadratic Loss and Sandy Gradient Search techniques must be used.

Two-Link Arm Hardware

The two-link arm hardware has been upgraded and "tuned" over the past year to improve linearity in the drive train and increase the positioning accuracy. The elbow joint was stiffened and self-aligning bearings installed to eliminate the joint stiction problem reported in the last annual report. Low quality bearings located in the rest of the drive system were replaced with higher quality bearings, and the drive shafts and joint supports were remachined and reinforced with steel inserts to improve alignment. Currently, we are replacing the plastic drive belts with pinned steel cables, because the plastic belts introduced nonlinear torque disturbances and tended to jump the gears under high loads. The positioning accuracy has improved substantially and is now within the 1 mm. accuracy originally planned.

Other features were added to the two-link arm to improve its utility. A new infrared LED driven end-point sensor was developed for use on the two-link arm which does not require the use of an isolation hood. We achieved an improvement in signal-to-noise ratio of about 40 dB and now have a (measured) positioning accuracy of 1 part in 1000 over the range of the sensor. Thus, for an area 1 meter square, we get 1 mm. position sensing accuracy. We are investigating the patentability of the sensor. The sensor is being modified to be able to track several targets at once in anticipation of target tracking and cooperating arm tasks.

Another feature added to the two-link arm was optical encoders on each joint. An electronics board was developed which can derive rate as well as position from the encoders. (This electronics board is also being checked for patentability. Now that rate and position at each joint as well as at each motor can be measured, complete information on the dynamic state of the arm can be directly measured as well as estimated.

A vertical axis (or Z-axis) drive system was added to the end of the forearm to give complete three dimensional positioning capability. The Z-axis drive was designed to move a 1 kg. payload the full 15 cm. range of travel in less than 2 seconds. A pneumatic gripper was installed at the end of the Z-axis to pick up payloads of up to 1 kg. Adjustable inertia disks were added to the motor shafts to allow adjustment of the plant vibration frequencies.

The PDP 11/24 computer hardware was upgraded to give us 8 new digital to analog converters for a total of 12, and to give us 24 new analog to digital converters for a total of 32. The digital i/o board has been utilized to accept the optical encoder digital signals directly.

Two-Link Arm Software

The operating system of the PDP 11/24 was upgraded from RT-11 V4 to RT-11 V5 and the fortran compiler (our main real time programming language) was upgraded from DEC RT-11 Fortran IV to DEC RT-11 Fortran 77. This new software speeds up our real time control programs by a factor of two. Thus, we can test more sophisticated controllers running at faster sampling rates. Also, some networking software was obtained that allows us to upload and download programs, experimental data, and even executable files between the PDP 11/24 and the VAX 11/782 which runs most of our control analysis

and simulation tools.

A basic library of real-time control and robotic routines has been developed which includes device drivers for the sensing and actuating hardware, manipulator kinematics (the so-called "arm solution"), and the manipulator Jacobian (differential joint/cartesian space relations). Other tools in development include automatic sensor calibration routines and automatic system identification programs. Nonlinear and linearized simulations of the two-link arm are being written for testing of new control schemes on the VAX 11/782. See Appendix J for an outline of the derivation of the equations of motion and simulation programs.

The most sophisticated controller yet developed for the two-link arm is one with the noncollocated end-point sensor and the motor tachometer rate signals being fed back through constant gains. This is a classical successive loop closure design. Integral control was added to compensate for the striction in the joints (which has been substantially reduced by hardware improvements).

Initial Two-Link Arm Controller Design Using End-Point Sensing

This first control design for the closed-loop end-point control of the two-link arm uses classical successive loop closure techniques. This design also assumes that the two-link arm dynamics can be decoupled and linearized so that simple single-input, single-output design techniques can be used. This is *not* the optimal or final control system design, but does allow us to test the end-point control concept on the two-link arm.

Assuming that each link is decoupled and the flexible vibration frequencies remain constant, successive rate and position loops can be closed around each joint motor. First, the rate loop is closed around the collocated tachometer signal from the motor itself. This gives the arm additional damping. Then the position loop is closed using the end-point measurement.

The X-Y cartesian position must be transformed into joint coordinates using inverse kinematics, the so-called "arm solution," and the differential X-Y cartesian position error signal must be transformed to joint position errors by the manipulator Jacobian. Additional integral control to compensate for striction at the motors and drive shafts was also added.

The block diagram for this controller is shown in Figure 3-5.

BLOCK DIAGRAM: Classical Controller for Two Link Arm

Using End-Point Sensing

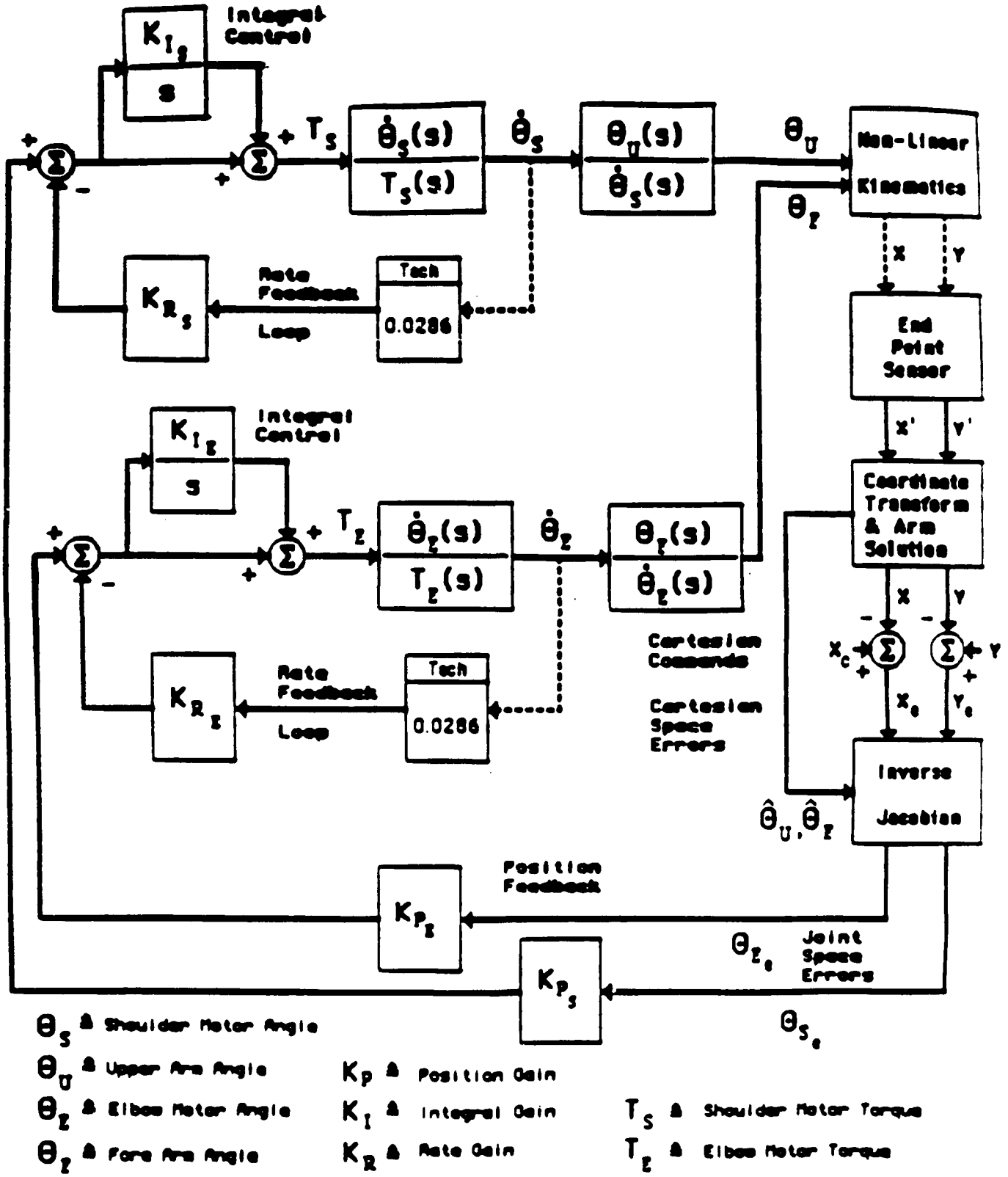


Figure 3-5

c. Strategies for Task Command and Control of Two-Link Manipulators

Two separate, rather extensive contributions were made to this area during the year by Bruce Gardner and Avi Weinreb. Both were sponsored by this AFOSR Contract, and both culminated this year in Ph.D. theses which are available as SUDAAR Reports, Refs. 6 and 7.

Gardner's early results were reported in Ref. 2 (pp. 41-46). An update summary is reported here:

Strategies for Moving Target Capture

An analytic design study is conducted to demonstrate circumstances under which the inclusion of feedforward compensation in a target-tracking control scheme can be expected to offer significant performance gain (e.g. enough to justify cost of implementation). In particular, a target-tracking controller design problem for a mechanical arm is developed to assess quantitatively the capacity of feedforward to provide a quicker, more accurate tracking response over wide ranges of uncertainty or variability in the dynamic parameters of *both* plant and target.

The Stanford Aeronautics and Astronautics Department Robotics Lab two-link, two-actuator mechanical arm, inherently a system with variable kinematic and dynamic parameters, provides an appropriate framework for this study. Using recent developments in the theory of quadratic synthesis of robust, low-order "optimal" controllers, control logic is developed – both with and without feedforward – that enables the arm end point to track a physical target characterized in part by periodic motion of variable or uncertain frequency and phase.

It is shown that, using relatively noise-free measurements of target position coordinates only, feedforward compensation can be expected to provide substantial reductions in tracking errors for given constraints on control effort, particularly when the range of variation in target frequency is large. As noise levels in the position measurements increase, the relative improvement in tracking accuracy (for a given level of control effort) offered by feedforward decreases. However, if target rate coordinates are also measured and used in the feedforward control scheme, the improvement is shown to be considerable even for fairly high noise levels in all target measurements.

Experimental verification of the predicted results contained herein is scheduled for the near future.

Strategies for Optimal Control with Multiple Bounded Inputs

Several aspects of optimal control of systems with multiple bounded inputs are considered.

A connection between the structure of multi-input systems and the existence of minimum time solutions with singular controls is identified. It is shown that systems which are decoupled or one-way coupled may have (depending on the boundary conditions specified) some singular, non-unique control components as part of the minimum-time solutions.

A new algorithm, which extends the basic gradient algorithm to optimal control problems with bounded controls is presented. When necessary, the resulting controls asymptotically use all the available control range, without violating the control bounds. The algorithm uses an adjustable control weight to enforce the control bounds.

Using the new algorithm, the minimum-time control problem of a two-link robot arm is solved, using an exact rigid body model and bounded controls. The problem is solved both for the given end-points case and for the new given-distance/unknown end-points case. The solutions provide new insight into the dynamic characteristics of the robot arm, pertaining both to path planning and design specifications.

d. Adaptive Control

This section describes new research that we are starting which we believe will allow us to make future contributions to robotics that will be fundamental and of central importance in the future. The leaders of this research are Daniel Rovner (an NSF Fellow) and Michael Sidman (a Digital Equipment Corporation Fellow).

Rationale

The development of high-performance control systems for complex, flexible mechanisms such as robot arms depends on the availability of a good mathematical model for the system to be controlled. This presents difficulties, since various important parameters influencing system dynamics may change during the course of system operation. An obvious example is a robot arm picking up a payload, the weight of the payload can significantly alter the dynamics of the arm. To achieve the best possible performance from the control system for such a mechanism, it is necessary to identify changes in significant system parameters *while the system is in operation.* The identified changes can then be used to update the controller design, again while the system is in operation, in order to optimize performance. This is the essence of adaptive control. Under this Air Force funding we have undertaken the development and testing of such control systems for physical systems available in the manipulator research laboratory.

System Description

As a first step, we have developed algorithms for adaptive control of a physical system previously built for the purpose of investigating problems in the control of flexible structures with noncollocated sensors and actuators. This system consists of four steel discs connected to a central torsion spring. RVDT sensors measure the angular displacement of each disc, and a brushless DC torque motor mounted on the second disc from the top provides the control input. The polar moment of inertia of two of the discs can be abruptly changed while the system is being controlled. This is accomplished by making two of the discs in pieces, as a series of concentric rings. Lifting off rings in succession causes significant and abrupt changes to occur in the dynamics of the system, providing a good test for an adaptive controller.

Accomplishments

The basic algorithm we have studied is the Self-Tuning-Regulator, which consists of an on-line identification scheme for measuring system parameters and a pole-placement algorithm for modifying the controller based on the identified parameters.

We have taken two approaches to the identification problem. In the first approach, the transfer function of the system is regarded as (almost) completely unknown, a priori. A recursive least-squares calculation is used to fit a system transfer function to data measured during controller operation.

The second approach regards the system transfer function as being completely known except for the inertias of the two afore-mentioned discs, which are regarded as being free to take on a continuum of values. The latter technique may have advantages in robustness and speed of convergence, since it involves less uncertainty about the physical system. However, the identification algorithm involves an algebraically nonlinear least-squares fit.

Both algorithms will be evaluated on the four-disc system, and the experimental results compared, beginning in year three just starting.

Technical Update on Task 4
INTEGRATED TACTILE SENSORS

Overview

The development of a tactile sensor for robotics applications has progressed in all the subtasks described in the First Annual Report. In addition, the importance of simple and accurate calibration of the final product has been appreciated. This appreciation has led to the design of a unique calibration procedure for the tactile sensor utilizing self-calibration data produced by the tactile array element. Major advantages of this calibration procedure include real-time compensation and calibration of the sensor output, the decoupling of sensor accuracy and bandwidth permitting both high accuracy and high bandwidth, direct telephone-link calibration of the in-factory industrial robot by the tactile sensor manufacturer, and instantaneous recalibration of the industrial robot by unskilled personnel when tactile sensors are exchanged.

a. Transducer Micromachining

The laser processing of glass and silicon for the micropackaging of the tactile array element has been investigated. CO_2 laser-drilled vias through a pyrex glass wafer to a metalisation pattern on a bonded silicon wafer are critical to improving the long-term stability of silicon pressure sensors. During the Second Year, electrical continuity for electrical connection to the array element without compromising either hermeticity or power consumption has been achieved. These vias are less than 100 microns in diameter with a contact resistance of less than 10 ohms.[1]

b. On Board Signal Processing

The deficiencies of the first version of the tactile array element on-board circuit, CPT3A, were described in the First Annual Report. Since then the effort on this subtask has been to improve the temperature and supply dependencies of the circuit. The revised circuit, CPT3B, has been designed, simulated, laid out, fabricated, and partially tested. This effort has been extremely successful. Without sacrificing oscillator stability of 230 ppm, which corresponds to a dynamic range of more than 4000, temperature dependence has been reduced from 1500 to 35 ppm/°C. This accuracy in an integrated circuit is unprecedented and has been achieved previously only by means of thin film components and individual laser trimming of much larger and more power consuming hybrid circuits. In addition, the principal source of supply dependence has also been eliminated, but as of this writing the substantial improvement expected in performance has not been measured.[2]

c. Array Multiplexer

This device, which controls and multiplexes the array as described in the First Annual Report continues in its circuit design stage. The digital logic design of the IC has been completed and is being verified using the Salogs logic simulator. The I^2L implementation of the logic has been completed also. Currently, work is progressing on the circuit imple-

mentation of the multiplexing electronics. For this purpose, a modified form the the EFL logic family is being investigated.[3]

d. Calibration

Inadequate calibration was recently identified (Open Discussion, IEEE Solid State Sensors Conference, Hilton Head, SC, June 6-8, 1984) as one of the two main factors retarding the penetration of silicon sensors into the commercial sensor market. Typically, commercially available silicon sensors are uncompensated for temperature effects and calibration for the sensed parameter must be performed on-site by skilled personnel employed by the user. (The second factor identified was inadequate packaging.) To solve this calibration problem, a novel calibration scheme has been developed for the tactile sensor. This scheme utilises thermal and manufacturing variability information multiplexed with the pressure data in the output of every tactile array element. The scheme is simplified compared to the usual computational methods in that it utilises a table look-up algorithm for both calibration by the manufacturer and operation in the factory. This algorithm requires only a uniqueness relationship between tactile sensors and reference temperature and pressure instruments in the manufacturer's calibration apparatus. Pressure is never calculated directly from the tactile sensor outputs during operation. Instead, these outputs merely point to reference instrument data stored during calibration. A major advantage of this scheme is that sensor accuracy and bandwidth are decoupled. Both are limited independently only by calibration table memory space. (It is to be noted that, since semiconductor memory is annually becoming larger and less expensive, this is no fundamental limitation.) Since no computational algorithm is involved in measuring pressure, the customary tradeoff between accuracy and computational time, that is, bandwidth, is eliminated. Furthermore, implementation of the calibration table memory space in EEPROM (electrically erasable programmable read only memory) permits tactile sensor calibration data to be transfered directly from the manufacturer's calibration laboratory to the in-factory industrial robot by telephone link. It also permits the in-factory robot to be recalibrated instantaneously by unskilled personnel when tactile sensors are exchanged (when operating pressure ranges are changed, for instance) simply by flipping a switch.

An automated apparatus for performing this calibration procedure has been designed and the prices of components obtained. The apparatus combines state-of-the-art pressure and temperature controllers for more accuracy than that achieved by calibration laboratories of current silicon pressure sensor manufacturers. This high degree of accuracy will enable us to observe the effects of mechanical, electrical, and packaging design changes upon sensor performance. It will also reduce the cost, time, and skill required to perform the calibration. It is expected that as the tactile sensor developes from prototype to production, this apparatus will be expanded from a small-lot to a large-lot capacity so that large dataset and throughput problems can be solved for future tactile sensor manufacturers.

Appendix A

PUBLICATIONS

General:

1. Air Force Contract No: F49620-82-C-0092.
2. AFOSR First Annual Report of the Center for Automation and Manufacturing Science, Stanford University, November, 1983.
3. Darpa First Annual Report on End Point Control of Flexible Robots, Stanford University, May 1984.
4. Darpa Second Annual Report on End Point Control of Flexible Robots, Stanford University, October 1984.

Task 3:

3-1 Rosenthal, D. and Cannon, R.H., "Experiments in Control of Flexible Structures with Noncolocated Sensors and Actuators," *Journal of Guidance, Control, and Dynamics,* Vol. 7, Number 5. Sept.-Oct. 1984.

3-2 Cannon, R.H., and Schmitz, E., "Control of a Flexible Manipulator Link," Proceedings of the 1981 Joint Automatic Control *Conference*, Charlottesville, Virginia, June 17-19, 1981.

3-3 Cannon, R.H., and Schmitz, E, "Initial Experiments on the End-Point Control of a Flexible One-Link Robot," *The International Journal of Robotics Research,* Vol. 3, Number 3. Fall, 1984.

3-4 Ly, U., "A Design Algorithm for Robust, Low-Order Controllers," Stanford University, SUDAAR 536, November 1982.

3-5 Ly, U., Bryson A., Cannon, R.H., "Design of Low-Order Compensators Using Parameter Optimization," to appear in *Automatica, The Journal of the International Federation of Automatic Control.*

3-6 Gardner, B. "Robust Feedforward/Feedback Control Logic for a Target-Tracking Mechanical Arm," Stanford University, SUDAAR 537, December 1983.

3-7 Weinreb, A., "Optimal Control with Multiple Bounded Input," Stanford University, SUDAAR 544, October 1984.

Task 4:

4-1 L. Bowman, J.M. Schmitt and J.D. Meindl, "Electrical Contacts to Implantable Integrated Sensors by CO_2 Laser-Drilled Vias through Glass," Workshop on Micromachining and Micropackaging of Transducers, Cleveland, OH, November 7-9, 1984 (Accepted for Publication).

4-2 M.J.S. Smith, M.A. Prisbe, J.D. Shott and J.D. Meindl, "A Micropower IC for a Capacitive Pressure Sensor," IEEE International Solid-State Circuits Conference, New York, NY, February 13-15, 1985 (Submitted for Publication).

4-3 F.B. Shapiro, J.D. Shott and J.D. Meindl, "A Custom IC for Multichannel Telemetry with Digital Sensors", Proceedings - IEEE Sixth Annual Conference IEEE Engineering in Medicine and Biology Society, Los Angeles, CA, September 15-17, 1984, pp. 715-718.

Appendix B

PERSONNEL

Principal Investigators:

Robert H. Cannon Jr., Professor and Chairman, Dept. of Aeronautics/Astronautics
Thomas O. Binford, Professor (Research), Dept. of Computer Science
James D. Meindl, Professor of Electrical Engineering and Codirector, Center for Integrated Systems

Center Consultants:

Daniel B. DeBra, Professor of Aeronautics and Astronautics
Gordon S. Kino, Professor of Applied Physics
Gene F. Franklin, Professor of Electrical Engineering
Lambertus Hesselink, Assistant Professor of Aeronautics and Astronautics
Victor Scheinman, Vice President, Automatix, Inc.

Task Area 2 Research Team:

Oussama Khatib, Post Doctoral Fellow
Yoram Kirson, Visiting Scholar
Ernst Triendl, Visiting Scholar
Avraham Rousso, Visiting Scholar
Ted Selker, Research Assistant
Joel W. Burdick, Research Assistant
Cregg Cowan, Research Assistant
H. David Goering, Research Assistant
Cary Gray, Research Assistant
John Clayton Hake, Research Assistant
Hong-Seh Lim, Research Assistant
Niels Mayer, Research Assistant
D. Sathyanarayanan, Research Assistant
Nancy Paulikas, Research Assistant
Craig Rublee, Research Assistant
Richard L. Vistnes, Research Assistant
William Wells, Research Assistant
Ron Goldman, Post Doctoral Fellow
Thomas Veatch, Research Assistant

Task Area 3 Research Team:

Wen-Wei Chiang, Research Assistant
Michael Grant Hollars, Research Assistant
Ross Koningstein, Air Force Training Grant Fellow
Ray Kraft, Air Force Training Grant Fellow
Arie Maharshak, Research and Development
Dennis Moore, TRW Fellow
Celia Oakley, NSF Fellow
Lawrence E. Pfeffer, Research Assistant
Daniel Rovner, Air Force Training Grant Fellow
Stanley Schneider, Research Assistant
Gad Shelef, Research and Design Engineer
Michael Sidman, Digital Equipment Corporation Fellow
Marc Ullman, Air Force Training Grant Fellow
John Wilson, Research Assistant

Task Area 4 Research Team:

Lyn Bowman, Research Assistant
Joseph M. Koepnick, Research and Development Engineer
Frederick B. Shapiro, Research Assistant
Michael J. S. Smith, Research Assistant

Appendix C

DISTRIBUTION OF THIS REPORT

AFOSR

U. Michigan

SRI

NASA (Dr. Larson)

DARPA (Dr. Kessler, Dr. Isler)

Dr. Thomas Walsh

Appendix D

GRAPHICS AND PREDICTION FROM MODELS

Richard Scott

Abstract

The first part of this paper derives algorithms and complexity arguments for a mainly implemented symbolic graphics scheme based on ray tracing arguments. The main step converts the scene definition into a 3D topological representation as seen from the viewpoint. It starts with a scene composed by unions, intersections and negatives of parameterized primitive volumes. Then limbs are found on the surfaces to given accuracy, and mutually consistent T-junctions and cusps are found in their projections to the image plane. The surfaces and image planes are represented by quad-trees. From these, the topology is extracted, represented by a "tooth-pick" structure which orders conical volume chunks that project into the same area of the image. The foremost regions can be displayed for hidden surface graphics. The topology of limbs, cusps and Ts is invariant over a range of viewpoints and models. Thus the tooth-pick model can be used as a framework for predictions. The second part of the paper examines changes in the topology as a function of viewpoint and surface shape, and partially derives geometrical representations for the changing sets of predictions.

Introduction

The main result of this research is a near optimal and mostly implemented graphics scheme that knows qualitatively what it is displaying. A scene defined by set operations on parametrized Generalized Cylinders, is converted into a 3D geometric representation as seen from the viewpoint. This representation and the method by which it is derived both have natural generalizations which analyse and support algorithms for further problems.

To put this in a broader context: The overall goal is to recognize objects in an image or industrial setting. This is done by searching through the viewing space and the space of modelled objects to find a match with the image. The search should be supported by geometric frameworks which organize the models and their characteristic views. Then the search is guided through the frameworks by features extracted from the image which are constant over ranges of viewpoints and models. So an important part of recognition is the job of starting with a model and then deducing what features will be quasi-invariant at different resolutions; in other words "prediction".

Past work, such as ACRONYM, has found many useful predictions, eg. trapezoid ribbons and attachments. But their power is limited by a lack of geometrical and topological structure in the way that the predictions are combined and used. This is a problem that our research hopes to solve.

Consider the function F, which lumps together modelling, prediction and graphics,

F: (Object-Model, Viewpoint, Ray-thru-viewpoint) ⟶ Intersections along the ray with the model.

F can be decomposed into three levels of abstraction, with each level factoring out invariants to be used as input for the level above.

1. Symbolic graphics function G with fixed model and viewpoint,
   G: Ray ⟶ Ray intersections

2. Prediction function P with fixed model,
   P: Viewpoint ⟶ Invariants of G
3. Deformation function D with varying surface shape,
   D: Model ⟶ Invariants of P

The functions G and P are a natural way of factoring the intersections of a model with the 4D space of arbitrary rays. In all cases, a method of getting at the invariants is to find and represent (a) the singularities and then (b) the continuous spaces in between.

1(a) For the function G the singularities are limbs and cusps (local) and T occlusion junctions (non-local). They are derived and shown to give an efficient hidden surface scheme by a sequence of complexity arguments based on ray tracing.

The singularities are found using three main components. First there is the underlying object modelling system, which consists of unions, intersections and negatives of the parametrized primitive volumes. The primitives are functionally defined Generalized Cylinders. Surface coordinate patches of the volumes are represented by quad trees. They are used to store surface information such as the limbs and the intersection lines with other surfaces. This allows limbs to be related across surfaces and followed across different primitives volumes. The intersection lines should be calculated automatically but at present they are given by hand.

Secondly, limbs and cusps are found. The algorithm repeatedly extrapolates a new guess at a limb point which is Newton-Raphson iterated to within error bounds. The extrapolation is based on the limb curvature and the step length is chosen to make the expected number of iterations constant (3 or 4). This gives minimum solution time with uniform accuracy; step length varies inversely with step angle.

Thirdly, T junctions are found by using an image plane quad tree. There is a T junction refinement algorithm which can recover from aliasing errors.

1(b) The singularities divide the surface into maximal patches that project (1—1)and which are ordered by distance from the viewpoint. The patches are extracted from the surface quad trees and must satisfy some simple Euler-like properties which guarantee physical consistency. This is important because the singularities are solved locally, approximately and must fit together globally. From these a "tooth-pick" graph is extracted. Each node is a conical chunk of volume bounded around the sides by rays through the viewpoint and at front and back by patches of surface. The chunks are lined up away from the viewpoint as if speared through by tooth-picks. The graph has the following properties which should make it a useful tool.

- Viewpoint centered information is made explicit. The graph structure is invariant over a range of viewpoints and model deformations: it constitutes a coarse level of prediction.
- It provides a framework for storing other predictions.
- The volume chunks can be formed by sweeping the 2D surface patch from front to back along the rays. This Generalized Cylinder type decomposition enables geometric reasoning between different dimensions.
- The inside and outside of the volume are represented uniformly.
- The derivation is clean cut and connects up to the next level of abstraction, the function P, which describes how predictions themselves can change.

One way of thinking of this is as an alternative modelling tool into which the Generalized Cylinder based model definition can be converted to support particular algorithms. Another useful representation would be something like an oct-tree, oct-graph, polyhedron hybrid.

2(a) The function P has three types of singularity, corresponding to chages in the topology of limbs, cusps and Ts, each of which takes the form of a generating line (L) on the surface of the modelled object with a direction vector (V) in the tangent plane at every point of L. A

ray parallel to V is swept along L, forming a ruled surface which divides up the viewing space. When the viewpoint crosses one of these surfaces a node of the tooth-pick prediction structure changes.

2(b) So the geometric framework for the prediction function P will be a graph of the viewing cells. Throughout each cell a prediction node of the tooth-pick graph is invariant. More work is needed to clarify the graph structure and the algorithms that produce it.

3(a) The Deformation function D is still in a sketchy state of development, though some of its singularities are known.

3(b) It has the potential to provide a flexible modelling hierachy.

An alternative decomposition of the function F may support recognition based on planar light stripes instead of images.

R: Ray $\longrightarrow$ Ray intersections with the model

R is a combination of the functions P and G; it takes a fixed model and allows the ray to vary over the 4D space of arbitrary rays. It can be decomposed analogously to G and P to give a wedge based tooth-pick model.

The rest of the paper is organized as follows:

Part One

1.	Singularities of the function G
1.1	Define some terms
1.2	Ray tracing complexity arguments
1.3	Algorithms for G part (a)
1.3.1	Limbs and cusps: search, follow, store
1.3.2	T junctions: find and refine
1.3.3	Modelling: 3,2,1,0D, outline, details
2.	Complete representation of the function G
2.1	Tooth-pick structure: symbolic graphics, prediction framework
2.2	Algorithms for G part (b)
2.2.1	Hidden surface algorithm
2.2.2	Image accuracy and display refinements

Part Two

1.1	Prediction singularities
1.2	Viewing cell graph: animation, changing predictions
2.1	Deformation singularities
2.2	Deformation structures

Part One

1. Singularities of G

1.1 Definitions

Some geometrical terms are defined.

A *limb*, also called a *contour-generator* is a closed loop of points on the surface being looked at, where the line of sight is tangent to the surface, i.e., perpendicular to the surface normal. It is directed so that the forward face lies on the left. The limbs divide the surface into forward and backward facing regions compared to the eye.

A *contour* is the projection of a limb to the image plane.

A *T-junction* is a point in the image where two contours cross. There are two different limb points on the same line of sight, the one closer to the viewpoint occludes the other.

At a limb point, let F be the line of sight vector, L the limb tangent vector, and N the outward surface normal. Both F and L lie in the surface tangent plane.
If $F \vee L$ is parallel to N then the forward face is closer to the eye than the back face, giving an *outer* limb point.
If $F \vee L$ is anti-parallel to N, the reverse is true, giving an *inner* limb point.
If $F \vee L$ is zero there is *cusp* point. F parallel to L means that the tangent plane intersects the surface along F, which is the definition of an *asymptotic direction* in the surface. So L is parallel to an asymptotic direction in a hyperbolic (saddle shaped) region of surface. This is another way of defining a cusp. There are two types of cusp, with L pointing towards or away from the eye.
Fig. 1 shows an oblique view of a torus.

1.2 Ray Tracing Complexity Arguments

Ray tracing arguments give an intuitive derivation of the fact that the only singularities of G are limbs and cusps (local) and T junctions (nonlocal), which was proved in 1955 by Whitney. The sequence of arguments makes successive refinements to an obvious first attempt algorithm, and thus shows that an optimal hidden surface algorithm must find the singularities.

If e is the maximum allowed error in the image, l is the total length of the limbs, c the total curvature of the limbs, then it has complexity

$$O(n \log n), \text{ where } n \text{ has } O\sqrt{lc/e}.$$

First attempt algorithm:

> 1. Shoot out a conical bunch of rays from the eye. Calculate where they intersect the surfaces of the object being viewed, and order the intersections along each ray by distance from the eye.

Algorithm 1 is inefficient because neighbouring rays (image points) have the same ordering of surfaces along them, except when an intervening ray goes through a limb i.e., is tangent to the surface there. So the order of surface regions from the eye along an arbitrary ray can be recovered from the ordering along the subset of limb rays.

> 2. Find the intersections of limb rays (tangent to some surface) with other surfaces in their paths.

This is inefficient because between neighbouring limb points the surface orderings can only change in two ways. When an intermediate contour ray passes through (i) a T-junction, (ii) a cusp.

3. First find the limb rays then find the subset of T junctions and cusp rays and the intersections with other surfaces there.

However, to find the complete surface orderings along rays through cusp and T points, it is not necessary to solve for the intersections and then order them. Instead, the difference in surface orderings are already known:
(i) Over a limb, the forward and backward facing regions are inserted together at one position in the ordering. For an outer limb segment, the forward face comes before the back face, while for an inner segment, the back face comes first.
(ii) Over a cusp, the forward and backward facing regions, adjacent in the order, swap places.
(iii) Over a T, the two regions of the other T branch insert themselves.
This set of differences can be solved to give complete surface orderings for each image region, by propagating partial orderings over the image.

4. Find the singular rays at limbs, T junctions and cusps, then solve for the unique, complete surface orderings in each image region that satisfy the ordering differences at singular rays.

This is the hidden surface algorithm, a large part of building up a representation for the function G. Still unspecified are the details of
(i) how the limbs, cusps and T junctions are to be found,
(ii) exactly what surface regions are being ordered, (and how to ensure that the Ts and cusps of each region, solved locally to within an error bound, are glooally consistent.)
(iii) and how to propagate up the orderings.

As far as complexity is concerned, (ii) and (iii) are both
O(number of image areas between contours)
which is a measure of the scene complexity, of smaller cost than (i).

Let e be the maximum allowed error distance between the approximate and exact contours in the image, l be the total length of the limbs, c be the total curvature of the limbs, and n be the number of limb rays solved for.

The average step length between adjacent rays is l/n, the average step angle is c/n, and the interpolation error is O(step-length $\times$ step-angle) so, $O(lc/n^2) \leq O(e)$, and n can be $O(\sqrt{lc/e})$.

The cost of solving for all n contour rays is $O(n)$. The cost of finding each cusp is $O(1)$; just see if the limb to ray angle has changed sign. To find T junctions, the n contour segments have to be tested for intersection with each other, costing $O(n \log n)$. (Assuming that no topological information is being used to direct the search.)

This results in the complexity of an optimal hidden surface algorithm, $O(n \log n)$, where n is $O(\sqrt{lc/e})$.

Actually the $O(n)$ limb solving dominates.

1.3 Algorithms for the Singularities of G

1.3.1 Limbs and Cusps

1. Search for a point on a new limb.

This is not a time consuming step; the algorithm steps along a fixed parameter path, and works out whether successive surface normals point towards or away from the eye. When they point oppositely, a limb passes between them. Has that limb already been solved? If so continue searching, otherwise give the two spanning points to the following algorithm.

2. Follow the limb over the surface, point by point, until it loops up with itself. This is the most expensive part of the whole hidden surface scheme.

Each limb point is actually a pair of points which span the limb closely. At the end of a partly solved limb we know the tangent direction and the previous step angles, (see fig. 2) so an initial guess to start a new iteration for the next limb point can be made by extrapolating the curve some distance. If the step length is too long, the Newton-Raphson search may not converge, if too short, unnecessary extra points are found. It is chosen to give the constant, optimal number of iterations (3 or 4) that gets a bounding pair with angles between their surface normals small enough to give an accurate estimate of limb tangent for the next extrapolation.

In other words, step length d, and step angle α along the extrapolation curve are chosen so that $d \sin \alpha$ is constant, which makes the initial error and number of iterations roughly constant.

If u and v are the two surface parameters, then the tangent direction in parameter space is, $du = d(N \cdot F)/dv$, $dv = -d(N \cdot F)/du$, where N is the surface normal and F is the line of sight vector. This and the previous limb point are used to fit an extrapolating spline to the limb in parameter space that has linearly changing curvature.

For planar surfaces tangent to the line of sight, the limb is taken along the front edge, so that the plane belongs to the back facing region. To find cusps on limbs

check for change in sign of the triple scalar product,

$$\text{Surface-normal} \cdot (\text{Line-of-sight} \times \text{Limb-tangent})$$

1.3.2 T Occlusion Junctions

The image is represented by a quad tree, with clipping usually done around the borders of the unit square. When crossing contours are found in a quad square, they are initially refined until the ordering at the T is clear. The condition is that both ends of one segment must be closer to the eye than both ends of the other.

Refining T junctions

This algorithm, which refines T junctions to arbitrary accuracy, is based on bisection. A complication is that due to aliasing errors the wrong bisected half may be picked and the algorithm must search back.

1. Two contour segments intersect each other. The longer contour is bisected (or a halfway interpolation), as an initial guess to iterate to a new limb solution point. If one of the two new halves crosses the shorter segment, this step can be repeated.

2. But what if nether cross the shorter segment? Then one end of the short segment must be enclosed by the triangle formed by the long segment and its two halves. The contour at the enclosed end is extended until it exits the triangle. If it crosses one of the halves, 1. can be resumed. But if it crosses the long segment, then it re-enters the stack of triangles formed by previous successful bisections. The contour continues being extended (step by step) through the stack, until it exits over a bisected side, and can return to 1. If this never happens, it finally exits out of one of the original segments that formed a T in the quad-tree. In that case two quad-tree Ts can be merged away, see fig. 5; they were just an artifact of approximate solution methods.

1.3.3 Modelling: 3,2,1,0D

Defining 3D Models

Scene:	Objects positioned in scene frame.
View:	Scene transformed into eye frame.
Object:	A connected object is given by:
	Primitive volume, or
	Deformed sub-object
	(e.g. reflection, stretch, negative), or
	union of sub-objects, or
	intersection of sub-objects.
Primitive volume:	Parameterized volume,
	that is piecewise continuous
	and differentiable.
	e.g. Generalized cylinder,
	half space of a plane.

This is theoretically equivalent to having the whole scene surface parameterized in a number of coordinate patches. Volume models are used because they are often more compact and easier to describe.

Examples of deformations:

One wing of an aircraft can be modelled as the reflection of the other. A tube will be the intersection of the outer cylinder with the negative of the inner one. The deformation of objects can be non-linear. E.g. looking at the scene through a distorting lens.

E.g. Definition of a screw, formed by cutting out the helical threads.

(Union (screw-head head-frame)

(Intersection (screw-body body-frame)

(Negative (screw-threads-helix body-frame))))

2D Modelling

The image plane, and each coordinate patch of the surfaces have lines and regions represented by quad-trees. The lines are stored with step length roughly proportional to $\sqrt{\text{radius-of-curvature}}$, to give constant accuracy, as explained in section 3.2 E. The discrete version of this is, step-length $\propto \frac{1}{\sqrt{\sin(\text{step-angle})}}$, with a minimum length equal to the allowed image error. The quad squares are subdivided until the ends of a step are in different nodes. Example of use: Intersections of lines are found using the quad-trees.

1D Modelling

Lines are represented by 2-way list of knot points, containing curvature information, and intersections with other lines.

More Modelling

A. Preprocessing
B. Primitive Volume Details
C. Quad Trees

A. Preprocessing

Given an scene, object, or view definition, perhaps with a repeated sub-object deformed in different frames, the modelling system constructs an Object-tree, with each node describing a separate object instance, and the leaves being the primitive volumes. The nodes provide a place to store information about each object that varies for different instances. For example, sequences of linear transformations between coordinate frames are combined. Another example; at the leaves of the Object-tree for a view, the coordinate patches and limbs of the surface are represented.

There are two more things that the modelling system does before it can be used (e.g. by the graphics program). First it finds the common intersection lines on the sufaces of intersecting objects. At present this information is given by hand. Then it works out which surface areas of the primitive volumes of intersecting objects actually lie on the surface of the composite object. It implements the formulas:

$$\partial(a \cup b) = (\partial a \cap (\text{Neg}b)) \cup (\partial b \cap (\text{Neg}a))$$

$$\partial(a \cap b) = (\partial a \cap b) \cup (\partial b \cap a)$$

where *Neg* means negative volume, to decide which side of each intersection line is on the outer surface. The intersection lines are stored in the coordinate patch quad-trees with the outer surface on the left.

B. Primitive Volumes

The basic building blocks are simple parameterized volumes. They have three parameters, (t, s, r) with each volume point $P = P(t, s, r)$ having unique parameters. $\partial P/\partial t$, $\partial P/\partial s$, $\partial P/\partial r$ form a right handed (not necess. orthogonal) set.

P is continuous in its parameters, and piecewise differentiable. The surfaces of discontinuity (called *jump-surfaces*) must be planar in parameter space. The purpose of this restriction is that it makes it easy to interpolate where a line segment in parameter space has crossed a jump-surface and to work out the closest jump-surface. At present these surfaces must be parallel to a

parameter plane. To allow more general volume discontinuties and surface edges, arbitrary planar jump-surfaces could be implemented using a Kd-tree structure.

The bounds of the volume are defined by restricting the parameters t,s,and r to all lie in some volume of parameter space, typically the unit cube. So there is a mapping (position function P) from the unit cube of parameter space to real coordinate space.

Opposite faces and edges of the cube can be identified with each other, e.g. when one of the parameters is an angle in cylindrical or spherical coordinates.

The remaining faces of the cube map onto the surface of the volume. They have separate quad-tree representations where they store special points, regions, and lines, such as limbs or curves of parabolic points. Each face will form a coordinate patch with one parameter fixed, and the other two parameterizing the surface. This is true for reasonable parameterizations.

So there are two kinds of edges on the volume surface,

1. where patches meet.
2. where jump-surfaces (derivative discontinuities) intersect the parameter cube.

C. Quad Trees

The condition for splitting squares up is that the opposite ends of each line segment stored lie in different areas. When testing for crossing of different segments, the longer one is split up further to the level of the shorter.

At the nodes, the segment end points are ordered around the borders of the square. This means that n segments in a square can be tested for intersection with each other in $O(n)$ time.

Also, regions of the tree can be marked by stepping around adjacent squares, moving away from the inside of the directed border segments.

2. Complete Representation of G

2.1 Tooth-pick Structure

The volumes in the scene are decomposed into conical chunks, convex with respect to the viewpoint, which are bounded around the sides by limb rays and at front and back by regions of surface. The connections of the chunks are represented by the "volume graph". Occluding chunks can be thought of as being lined up along tooth-picks emanating from the eye and ending in an area of the image plane. This is represented by the "image graph", which links up the smallest areas of the image plane undivided by projected limbs. Adjacent image areas have tooth-picks which differ by just one conical chunk.

The volume graph is extracted from the surface and image graphs, where the "surface graph" represents adjacent regions of surface which project (1—1)to the image. It in turn is extracted from the quad-trees of the surface patches.

2.2 Algorithms for G part (b)

2.2.1 Hidden Surface Algorithm

1. Split the image plane up into areas undivided by contours. Each area is bounded by a number of contour segments with Ts or projected cusps at the vertices. This is the "image graph" extracted from the image quad-tree. Since image plane is not here being used for any distance measurements, it might be better to think of it as a small sphere surrounding the viewpoint.
2. Split the whole surface up into maximal regions that project (1—1)to the image. *i.e.*, which do not twist around to hide themselves. These are sub-regions of the forward and back facing areas bounded by limbs. This is the "surface graph".
3. In each image area order the surface regions that project over it by distance from the eye. There seems to be an optimal algorithm to do this which for a single object without self penetration requires only the information provided by Steps 1 and 2 (see below).

For hidden surface graphics including transparency, the surface regions can be displayed in order, up to the first opaque one, for each area of the image.

Step Two

The limbs divide the surface up into alternating forward and backward facing regions called *faces* with respect to the viewpoint. Each face is bounded by a set of different limbs called a *limb set*.

The only way that a face can occlude itself is by having T junctions or cusps in its limb set. If there are none, then the face projects (1 −1)to the image.

Ts and cusps mean that the face needs to be split up further to form (1 −1)sub-regions. The splitting algorithm starts with an unaccounted for T or cusp and uses the image graph to follow the projection of the limb over the interior of the face until it reaches another T or cusp.

There are Euler-like relations governing the physically possible combinations of Ts and cusps. They can be derived by considering the ways that Ts and cusps appear and disappear as the viewpoint moves around (see Part Two). They can appear, regroup and disappear in a fish-tail (a pair of cusps and a T) or as a lone pair. The test to find cusps is local to a small patch of surface, and the T test is local to an area of the image. It is important that these local, approximate calculations can be combined into a globally consistent image. Fig. 5 shows two spurious T junctions. Fig. 6 shows the T junction of a fishtail detected without its cusps. As the accuracy constraints are relaxed, the resulting image should be the exact projection of a 3D volume that approximates the real one, degenerating into a sphere-like blob. The Euler-like relations can be

used to enforce this, (the details need to be worked out) so that, for example, the two undetected cusps of fig. 6 would be added. The false Ts of fig. 5 are already merged out by the T refinement algorithm.

Having the Ts and cusps consistent for each limb set separately, $\Longleftrightarrow$ the whole image is consistent.

Step Three

This last step orders the (1—1)surface regions that occlude each other and project over one node of the image graph. A partial order already exists from the T junctions, but surrounding contours must be resolved. One method of doing this is to intersect the surrouding surface with a limb ray from the other one. However this is rarely necessary. For an object with self penetration, or two disconnected objects that share no T junctions, exactly one ray intersection is needed.

A detailed algorithm has not yet been worked out, but the following is a sketch of a proof that the surfaces can be ordered pairwise, and hence completely.

Take a pair of (1—1)surface regions A and B that intersect in the image. There are two ways that they can be ordered immediately.

(i) if they have a T junction between the image projections of their boundaries.

(ii) if they share a boundary and are adjacent on the surface.

If that fails, find a connected path of regions from A to B through the surface graph.

Step down the path, starting at A, checking each region against B by (i) and (ii) to get an ordering.

These 3 steps outline the hidden surface scheme.

Some previous versions of these algorithms that found limbs, Ts and cusps without organizing the surface regions, have been implemented e.g., the *two views of a helix*, *figs.* 7,8. The program of interpreted maclisp found about 100 limb points, and took about 4 minutes.

2.2.2 Image Accuracy and Display Refinements

If $F1$ and $F2$ are position vectors of the two bounds for a limb point in the frame of the eye, and are distance d apart, and θ is the angle between their surface normals $N1$ and $N2$, then, $\text{sign}(N1 \cdot F1)$ is opposite to $\text{sign}(N2 \cdot F2)$.

The maximum possible image error bounded by e, is (see fig. 3)
$rk(1 - \cos\theta/2)$ where k is the ratio:
image-plane-distance/$\min(|F1|, |F2|)$,
and r is $d/2\sin\theta/2$.
This reduces to

$$\frac{dk\sin\theta/2}{2(1+\cos\theta/2)} < e \rightarrow d\sqrt{\frac{(1 - N1 \cdot N2)}{2}} < 2e/k$$

This is the condition for stopping the iteration that refines the bounds of a displayed limb point.

For a projected step of length l in the image, with step angle α, the error in any interpolation scheme will be $O(l\sin\alpha)$. α is a measure of the inverse of radius-of-curvature, so the error is $O(l^2/\text{radius-of-curvature})$. There are two alternatives. If the image interpolating error needs to be bounded by fixed error e, then the step length l between knot points should be $O(\sqrt{\text{radius-of-curvature}})$. However if the image interpolating just needs to look good, then the error should be proportional to step length. So $l\sin\alpha \propto l$, giving constant step angle, and step length O(radius-of-curvature).

Part Two

1.1 Prediction Singularities for Changing Viewpoint

There are three kinds of two dimensional singularities of the prediction mapping P, corresponding to changes in the topology of limbs, cusp and Ts.

Each takes the form of a generating line (L) on the surface of the modelled object with a direction vector (V) in the tangent plane at every point of L. A ray parallel to V is swept along L, forming a ruled surface which divides up the viewing space. When the viewpoint crosses one of these surfaces a node of the tooth-pick prediction structure suddenly appears or splits up.

Summary:

1. Limbs meet:
   Line of parabolic points, with asymptotic direction at each point.
2. Fish-tail appears:
   Line of asymptotic inflexions, with the inflecting asymptotic direction at each point.

3(i). Pair of Ts appears:
   Pair of T generating lines, where corresponding points, one from each line, have a common tangent plane. The special direction is a linking vector along the line joining corresponding points.

3(ii). 3 aligned T junctions – a star junction:
   Three generating lines where corresponding points are colinear. The special direction is the linking vector.

1. (a) Limbs in the same limb set (i.e. surrounding a forward or backward facing region of surface), touch and merge, joining up two regions that face the same way.

(b) A new limb loop is born at a point.

Both of these imply that the derivative
of (surface-normal · line-of-sight) is 0 in all directions at that point. The only way that this can be satisfied is to have one principal curvature zero, with its asymptotic direction coincident with the line of sight. So the singular viewing surface is ruled through the surrounding space, by sweeping a line tangent to the asymptotic direction along the closed loop parabolic curves on the surface.

Generically, one end of asymptotic direction points into the hyperbolic region, the other into the elliptic region. Looking along the asymptotic ray from elliptic to hyperbolic side, (b) is seen; the reverse direction gets (a).

2. Two cusps and a T junction appear in a fishtail shape, at a point on a limb in a hyperbolic region.

The condition for this is that the line of sight is tangent to one asymptotic direction at a point where the asymptotic line has an inflexion.

To derive this result, (see fig 1), imagine that the viewpoint is moving (with 3 d.o.f) so that the two cusps approach each other and merge. Since their limbs are tangent to the asymptotic direction, the cusps must approach by sliding in along that direction, which gives the asymptotic line an inflexion point where they meet.

So the singular viewing surface is ruled by sweeping a line in the asymptotic direction along the curve of asymptotic inflexions. This curve may intersect itself or intersect the curve of asymptotic inflexions of the other asymptotic direction, or be tangent to the curve of parabolic points. e.g. the curve around the middle of the hole in a torus.

3(i). A pair of T junctions appear at a point.

Two contours become tangent in the image, and then form a pair of Ts. In order to see this singularity, the two surface points, whose contours are tangent, must have coincident surface tangent planes, with the line of sight aligned through both points.

Pairs of T junction generating points form into two parallel closed loops. Each point on the upper loop has a corresponding point on the lower loop with a coincident tangent plane, and direction pointing to its mate. The singular surface is ruled by sweeping the direction around the loop.

3(ii) Three contours cross at a point in the image.

There are several interesting 1 d.o.f viewpoint singularities and some weird 0 d.o.f. ones, where the viewpoint has to be at one special point in space. See [1] (pages 145 - 149) for a complete classification.

An important local 1 d.o.f. case is the borderline between 1(a) and 1(b) where the asymptotic direction is parallel to the parabolic curve.

1.2 Viewing Cell Graph

A representation of the mapping P needs to be built up, based on its singularities. Just as the tooth-pick structure of G supports symbolic graphics and predictions over a limited range of viewpoints, this representation could provide a framework for general predictions and animation.

In [5], Koenderink and van Doorn describe a graph structure called the "Visual Potential" which predicts what an object looks like from its different characteristic views. Each node represents a volume in the viewing space, with a link between adjacent volumes. These volumes are sliced out of space by the ruled surfaces described above.

To be useful for recognition or animation, it is not necessary to generate the whole graph; it is enough that at any viewpoint the structure of the graph locally can be generated. This is analogous to not generating the whole image, but only being able to work out what the image looks like near a single ray. At a node, something like the "tooth-pick" structure could be stored; along with the likely singularities (3 different types), so that links can be generated in response to a change in viewpoint. E.g., see fig. 9 for the sequence of views of a "dumb-bell", moving from an oblique to an end-on view.

2.1 Deformation Singularities

Some of the local singularities are:

1(a) A parabolic curve apears at a point. e.g. stretching and bending a sphere.

(b) A parabolic line pinches off to form two parabolic lines. eg. turning a banana into a dumb-bell. This can only happen when the two points that approach each other where the split occurs both have asymptotic direction tangent to the parabolic line. The two points anihilate each other at the split. (They are a parabolic line analog of cusps on limbs.)

2. Lines of asymptotic inflexion appear and split. Their cusp analogs are points of self intersection of intersection with other curves of asymptotic inflexion, and where they are tangent to a parabolic line.

3. Similarly the T generators appear, merge and split.

2.2 Deformation Structures

The singularities produced by changing the surface shape suggest a model hierachy. One object might be described in terms of another by giving the sequence of singularities that occur when its surface is deformed to fit the other's. E.g., Forming a "dumb-bell" (see fig. 10) by deforming a sphere. First the sphere is bent into a banana shape when a parabolic curve, bisected by a curve of asymptotic inflexions appears at a point on the sphere's surface. Then the two ends of the asymptotic inflexion curve, which are also on the parabolic curve, meet up around the girth of the banana. The parabolic curve splits into two, and the banana becomes a dumb-bell.

References

[1] Arnol'd, V.I., "Singularities of Systems of Rays," *Russian Math Surveys*, **38:2** (1983), 87-176.

[2] Binford, Thomas O., "Figure/Ground: Segmentation and Aggregation," *Proceedings: Rank Prize Fund*, Conference in England, 1982.

[3] Brooks, Rodney A., and Thomas O.Binford, "Interpretive Vision and Restriction Graphs", *First National AAAI Conference*, 1980.

[4] Koenderink, J.J., and A.J.van Doorn, "The Singularities of the Visual Mapping", *Biological Cybernetics*, **24** (1976) 51-59.

[5] Koenderink, J.J., and A.J.van Doorn, "The Internal Representation of Solid Shape with Respect to Vision", *Biological Cybernetics*, **32** (1979) 211-216.

[6] Horning, Cristoph, "An Approach to a CalculationMinimized Hidden Line Algorithm," *Comput.& Graphics* **6:3** (1982) 121-126.

[7] Scott, Richard S.F., "An Algorithm to Display Generalized Cylinders," *Proc. IU Workshop*, April 1983.

Appendix E

STEREO MODELING SYSTEM: A GEOMETRIC MODELING SYSTEM FOR MODELING OBJECT INSTANCE AND CLASS

Jun Takamura and Thomas O. Binford

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

This paper describes an intelligent, user-friendly geometric modeling system which enables simple, fast building of 3D models for the Image Understanding system. First, an instance of an object is taught through the operations in which Generalized Cylinders are fitted to each part of an example of the object in 3D space formed by stereo images. Model descriptions of the instance are generated automatically after the operations. Second, class model of the object is learned inductively using the descriptions of the instances and model descriptions of the class are also generated. The generated model descriptions can be used as input of the ACRONYM system.

1. Introduction

As the research of model-based image understanding systems advances and these systems come to be applied to the recognition of various objects, an intelligent modeling system which can build 3D models efficiently is becoming necessary. Namely, a modeling system which enables simple, fast model building is required. Also, the system must be easy to learn in terms of how to build 3D models, and user-friendly from the viewpoint of the user-interface. A system for geometric modeling called Stereo Modeling System which satisfies these requirement has been designed and built.

In the ACRONYM system, 3D models were built through the text-based description language.[1] The problem was that model descriptions must be made by measuring various parameters, such as the size of a part of the object and the angle between parts, and by writing out long model descriptions. For example, to build a model of a relatively simple object like an aircraft, a couple of hundreds lines of model descriptions were required. It is not easy for a person who does not have enough knowledge of computers to handle the modeling language. Moreover, as the model descriptions are made by hand, errors in model descriptions are likely to occur. In order that a modeling system may be used widely and be applied to various objects, it is ideal that even a person who is not a computer expert can build 3D models of objects easily and without errors.

The 3D model has also been studied in the fields of CAD and Computer Graphics. However, the major difference between 3D models in these fields especially in CAD and in Image Understanding is that in the latter, objects to be modeled actually exist in the world. With this advantage, the system builds models of object instance and class from actual examples of the object through the following two successive stages.

1) A model of an object instance is taught interactively in 3D space formed by stereo images of an example and a stereoscope, i.e., geometric features of the object are specified directly in 3D space via the user-friendly interface, for instance vertex and curve, without symbolic descriptions, utilizing prior knowledge of the object.
2) The system learns the model of the object class inductively through the models of instances, and builds model descriptions of the object class automatically.

The model building of object instance is done by fitting Generalized Cylinders to each part of the example in 3D space, based on comparison of the displayed Generalized Cylinder and the actual object. The teacher divides the object into several parts according to the required accuracy of approximation, specifies relations between parts, and builds the model of the whole parts efficiently, making use of the descriptions of the parts which are already defined. The system calculates parameters such as the size of the part or the angle between parts and generates model descriptions of the object instance. Section 2 describes the details of how to build instance models. Class models are built by generalizing model descriptions which are expressed in terms of algebraic constraints. Class models are also specialized by adding new constraints obtained through misrecognition experiences. The details will be described in Section 3. Specifying geometric features of objects directly in 3D space and teaching instance model using an example of the object is the most intuitive way to build 3D models. Moreover, making errors in model descriptions can be prevented by the procedure in which a building 3D model is displayed and always compared to the actual example. In Section 4, the user interface is described in detail through the actual example. The generated model descriptions can be used as direct object models of the ACRONYM system.

2. Fitting Generalized Cylinders in 3D Space

3D models are usually built by using standard components like cylinders, parallelepipeds, spheres, etc.. 3D models can be constructed to any degree of accuracy if the number of the faces is increased.[2] However, tractability and simplicity rather than accuracy of the model is important in image understanding for the processing stages of prediction and matching. Generalized Cylinders[3] have been widely used as such powerful components for the 3D models of image understanding systems, for instance the ACRONYM. The Generalized Cylinder is also used as a primitive for building models in this system.

2.1. Generalized Cylinder Fitting Using Knots

The most intuitive way to build 3D models is to build models directly in 3D space viewing actual 3D examples. 3D input methods have also been studied in CAD and Computer Graphics.[4] However, in the case of 3D models of image understanding, actual objects usually exist. In this system, 3D models are built efficiently in 3D space formed by stereo images, based on actual examples. Namely, a stereo display of an example is viewed through a stereoscope and the Generalized Cylinders are fitted to each part of the object by specifying knots in 3D space. The occurrence of errors in the model description can be prevented by the continuous comparison of displayed Generalized Cylinder with the actual object.

As the geometric modeling of synthetic objectives requires a high degree of accuracy, many knots must be specified in order to fit models to curves or surfaces. Although the surface can be fitted with any accuracy by polyhedral approximation, input and modification of the model is not easy. The fitting of curves and surfaces by Bézier or B-spline formulations also requires many control points. However, describing objects in terms of Generalized Cylinders for the purpose of image understanding purpose can be done simply by the following method.

1) From knots to a cross section

A Generalized Cylinder is defined by a cross section which sweeps along a spine, changing shape according to a sweeping rule as it continues to sweep. The types of Generalized Cylinders used in the

ACRONYM is shown in Table 1.

Cross-section	Sweeping rule	Spine
circle	constant	straight
square	linear	circular
rectangle	bilinear	non-perpendicular
regular-polygon		

Table 1. Types of Generalized Cylinder

A cross section in a plane and a plane can be determined by three knots as shown by an example in Figure 1. All the types of cross section in the ACRONYM can be fitted by three knots as illustrated in Figure 2.

2) From cross sections to a Generalized Cylinder

As described, a Generalized Cylinder is formed by a cross section's sweeping along a spine. All the types of Generalized Cylinder in the ACRONYM can be fitted by three cross sections at most. For example, three cross sections are required for the types of Generalized Cylinder which have a circular spine as shown in Figure 3. Two cross sections are required for the types of Generalized Cylinder which have a straight or non-perpendicular spine and have a sweeping rule other than the constant type. Consequently, nine knots are required to fit three cross sections. However, the most common type of Generalized Cylinder, which has a straight spine and a constant sweeping rule, can be specified efficiently using only four knots as illustrated in Figure 4. Due to the smaller number of required knots compared to the usual surface fitting and the local controllability in the stages of cross section fitting, Generalized Cylinder fitting can be done simply and efficiently.

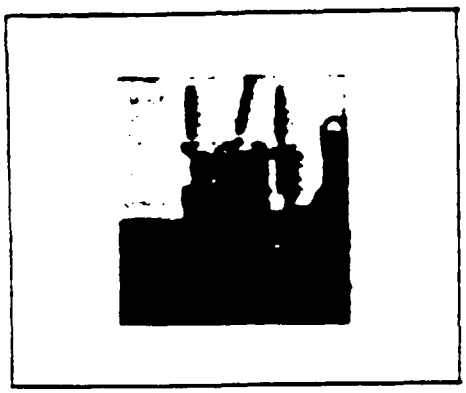


Figure 1: An example of fitting a cross section by three knots.

2.3. Using Pre-defined Parts Efficiently in Model Building

Some 3D objects have a hierarchical structure and have the same subparts. Once a part is described in terms of a Generalized Cylinder, the description can be used to define other parts efficiently. Besides, this has the advantage that 3D models can be described in compact form and modifications can be done to all the common parts simultaneously.

1) Identical part

The same definition of the Generalized Cylinder can be eliminated by the selection of an already defined part name. Also, if the relative position and the orientation of the part is identical to already defined parts, specifications of the relations described later are not necessary either. Otherwise, three knots are necessary

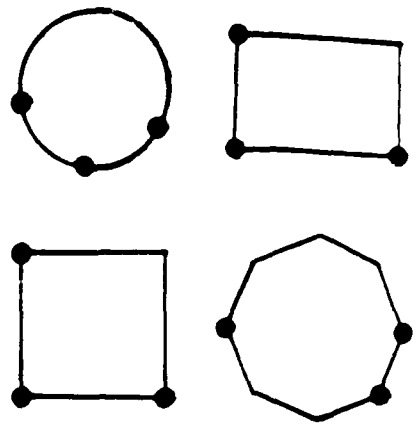


Figure 2: Cross section types

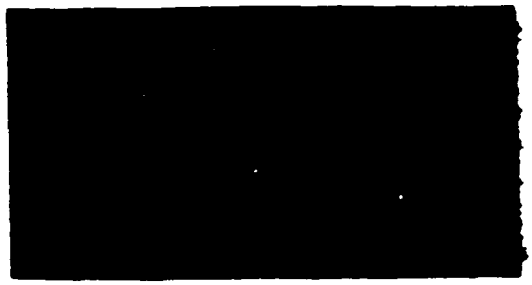


Figure 3: Fitting a Generalized Cylinder from knots.

for the specifications of the position and the rotation of its local coordinates system. Iterative definition of the identical parts enables the simultaneous definition of several identical parts. After the Generalized Cylinder definition of the first identical part and specifications of the number of iterative parts, two or three knots are specified to determine the positions of the rest of the identical parts. Namely, if the positions of the identical parts are circular, three knots are required. If the position is linear, two knots are enough to determine the positions.

2) Symmetrical part

Some objects have symmetrical subparts. The symmetrical part can also be described in terms of the model descriptions already defined. For example, the port wing of the object jet aircraft is symmetrical to starboard wing. In this case, the specifications of port wing can be eliminated by utilising the definition of the starboard wing which is symmetrical against the y-z plane of the fuselage. The model description of the port wing is generated from the description the starboard wing.

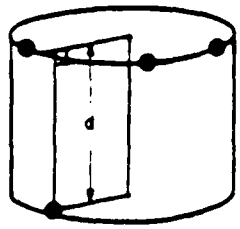


Figure 4: Fitting a Generalized Cylinder by four knots.

3.3. Spacial Relations between Parts

The final geometric relation between each part of an object is described in terms of the relative position and the relative rotation of the local coordinates system of the part to another part. The relations between parts described below can be specified through the menu selection. At least the affix relation must be specified to another part. The system automatically calculates necessary parameters for the transformation of coordinates system based on the relative location of the part in 3D space and generates relation descriptions. Also, several relations can be specified for the accurate definition of the geometric relation. For instance, relations like Align, Coplanar, and Flush can be specified to other parts which are already defined. If these kinds of relations are specified, the local coordinates of the Generalized Cylinder is slightly modified and the accurate model descriptions are generated.

Also the subpart relation must be specified, i.e. the name of the subpart must be defined when the subpart is specified. The system generates a subpart tree using the name. If the object example is not the first example of the class, the name can be selected from the menu.

3. Forming Object Class from Examples

After the model descriptions of the object instance are generated as described in Section 2, the model of the object class is learned inductively, based on the examples or models of the object instances.

3.1. Representation for Learning

In the ACRONYM system, data structures are represented by the frame-like structure, i.e., each data object is an instance of a unit. Units have a set of associated slots whose fillers define their values.[1] Objects are represented by object graph whose arcs are subpart and affixment. The subpart arc describes a coarse to fine structural hierarchy represented by a subpart tree. Affixment arcs relate coordinate systems of objects. Class is represented through a mechanism of constraints and a restriction graph. The constraint is inequalities on algebraic expressions which defines a set of values which can be taken by algebraic expression in the slot, i.e., $5.0 \leq width \leq 6.0$. The restriction graph is used to organize the constraints into class, subclass and instance. Namely, a set of constraints can define object class, subclass or instance and the hierarchy of each set of constraints can be defined by restriction graph. Figure 5 shows an example of the class hierarchy of the jet aircraft. The nodes in each level have corresponding constraints, though one set of subpart tree, affixment tree, cylinder descriptions exits for the object class. The constraints of the predecessor are applied to its successor nodes, for instance, the constraints of B-747 are applied to both B-747B and B-747SP.

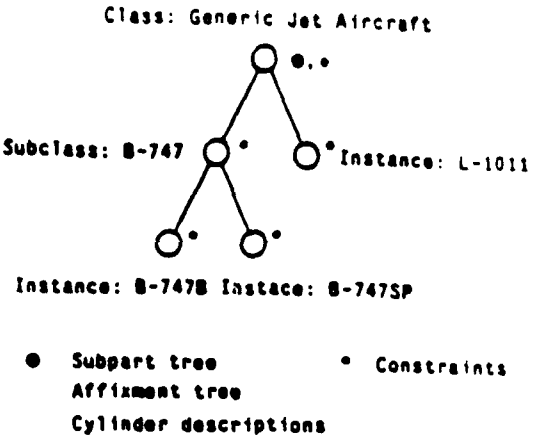


Figure 5: An example of the class hierarchy.

3.2. Rules of Generalisation

Generalisation to form a class model can be done simply owing to the representation described above. The model descriptions of the object instance built by the system as described in Section 2 are written in terms of constraints("=" is one of the forms of the constraints). The constraints can be used for variations in size, in structure, and in spatial relationships. For example, a structure can be expressed like LEG-QUANTITY = 3. Generalization of the model descriptions can be done by generalizing these constraint. For example, suppose B-747B has the constraint,

$$\text{FUSELAGE-LENGTH} = 67.3$$

and B-747SP has the constraint,

$$\text{FUSELAGE-LENGTH} = 52.0$$

These constraints are generalized and the following constraint of the FUSELAGE-LENGTH of the class B-747 is obtained.

$$52.0 - \epsilon \leq \text{FUSELAGE-LENGTH} \leq 67.3 + \epsilon$$

where ϵ is a tolerance. Constraints for structure can be generalized in the same manner. If the corresponding subpart cannot be found in a certain instance, the quantity of the subpart in that instance is regarded as 0.

In summary, the teacher teaches the object instance using an example and specifies the name of the class/subclass to which it belongs. The system generalizes the model descriptions of the object class as described. Namely, all the constraints in the predecessor nodes in the restriction graph are generalized. For example, if we add a new instance to the class B-747, say B-747X, the constraints in node 'B-747' and 'generic aircraft' are generalized.

The inductive learning described above uses only positive examples and the model descriptions sometimes become overgeneralized. The specialization of the class model through the negative examples or counterexamples should be done. In this system, descriptions of the class model can be specialized with the addition of new constraints. When the ACRONYM system misrecognizes an object which is not included in the class, the example can be used as a negative or counterexample. Using this misrecognized example, the teacher can add a new type of constraint in the model description.

Learning from examples can be categorized into two types, the one-trial, and the other, incremental. "Although this incremental method parallels human learning, it is apt to lead one down garden paths by an injudicious choice of initial examples in formulating the kernel of the new concept".[7] Therefore, in this system, the teacher must select good examples for forming good class descriptions. Finally, the system is currently being planned to include some type of constructive induction in order to increase its power.

4. User Interface and Examples

The user-friendly interface of building models are described using actual examples in this section. Namely, the method to input knots in 3D space and to fit the Generalized Cylinder is described, based on the Generalized Cylinder fitting method described in Section 2.

4.1. User-friendly Interface

Figure 6 shows the following hardware devices which are used as user interface. (1) Scanning Stereoscope ODSS III (2) display device for stereo images (3) Voice Recognizer SYS300 (4) trackball (5) TSS terminal. The system generates 3D models through the following user-friendly interface using three hardware devices.

1) Convenient Input

The system is designed to use only voice and the pointing device. The keyboards are not usually used except in the case where names

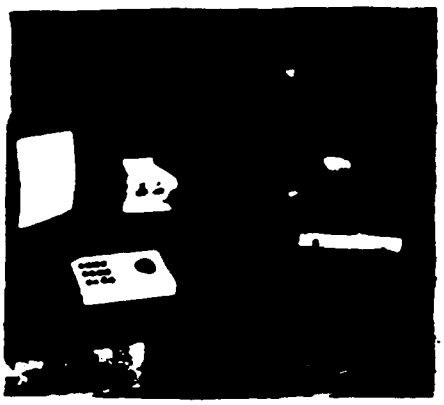


Figure 6: Hardware of Stereo Modeling System

are specified. The voice input is especially useful in Generalized Cylinder fitting operations in which both eyes are fixed on an object and a hand is placed on the trackball as described later.

2) Menu Selection

Commands to the system are done in terms of menu selection using voice. With this menu method, users need not memorize commands and the operations can be learned easily, guided by the system.

Also, short help messages explaining what to do next are displayed in the lower part of the screen whenever necessary. If the user is completely lost as to what to do next, "help" in the menu can be selected. The system always prepares help commands in menus and the user can read displayed help messages.

4.2. 3D Input

The knot input is done directly in 3D space formed by stereo images. The position of the knot in 3D space can be calculated by the corresponding points in two different images based on the theory of Photogrammetry.[3] From the viewpoint of speed and accuracy, 3D input is done through the following two stages.

1) Coarse initial positioning with two cursors

The coarse initial position of the knot is specified by pointing to corresponding points in the stereo images, using two cursors displayed on the same epipolar line[4] in the stereo pair. Although the 3D positioning using the cursor moved by the trackball is fast, it is impossible to specify the position between one pixel. The one pixel disparity between the corresponding points in stereo images sometimes affects the accuracy of the depth measurement significantly. Therefore, in order to position the knot in 3D space with a high degree of accuracy, the following 3D pointer, which can be moved in 3D space with any degree of accuracy, is used for fine positioning of the knot.

2) Fine positioning with 3D pointer

The accurate 3D position of the knot can be specified through the 3D pointer displayed in 3D space. The 3D pointer can be moved in six directions in 3D space with voice commands, as illustrated in Figure 7. Also, the speed of the 3D pointer can be changed, i.e., the speed can be changed to x2 and 1/2 by "fast" and "slow" commands respectively. For instance, if you say "slow" three times, the speed is reduced to $(1/2)^3 = 1/8$. The 3D pointer can be moved in 3D space after all the initial input of necessary knots for generating a cross section or a Generalized Cylinder is finished. Although it is difficult to display the 3D pointer between pixels in the image, the 3D position of the knot can be determined with great accuracy by observing a displayed cross section reflecting the move of the knot. For example, a radius of a circular cross section is sometimes highly sensitive to the slightest movement of one knot.

4.3. Stereo Images

Stereo images of objects are obtained by placing objects in front of a stereo camera. Currently, the online stereo camera is not used and stereo images which are stored in a disk file are used by the system. As described before, the one pixel difference is sometimes very important for accurate Generalized Cylinder fitting in 3D space, so a high-resolution display device is needed. To solve this problem, we built zoom function, i.e., a part to be modeled can be enlarged and displayed whenever accurate modeling is required. This zoom function enables accurate modeling, as if a high-resolution display device were available. Also it is better that the stereo images themselves should be high-resolution. For this purpose, in addition to the normal size picture, high-resolution stereo images are also used by the system. Any portion of the stereo images with any resolution are displayed using the resolution pyramid of the stereo images with this zoom command.

4.4. Examples

The user-interface described above will be shown with actual examples of Generalized Cylinder Fitting operations. Figure 8 shows a displayed original stereo pair of industrial parts with a command menu. Suppose we built a model of one of the parts. After the zoom command is selected and the location of the zoom window is specified with the trackball, the enlarged stereo pair of the selected part is displayed as shown in Figure 9. With two cross-haired cursors, the initial 3D position of a knot is specified as shown in Figure 10 after the type of Generalized Cylinder is selected from the menu. In this figure, "+" shows the position of already specified knots. After the specifications of three knots, the knots can be moved by selecting one knot with the cursor pointing to the vicinity of the knot. The 3D pointer "+" appears as shown in Figure 11 and the 3D pointer can be moved in 3D space with voice commands. The displayed cross-section shown in Figure 11 is also moved/transformed, reflecting the movement of the 3D pointer. After the fitting operation of the cross section is finished, the last knot, in this example, is specified. When all the knot input is done, the Generalized Cylinder is displayed and the knots can also be moved at this stage. Figure 12 shows the displayed Generalized Cylinder in terms of two cross sections. As this system does not use any special graphics hardware and since it is written in *Lisp* called *Soreemoo Lisp* (also some graphic packages are written in C), it takes too long for real-time interactions of moving/transforming Generalized Cylinders are well displayed with hidden-line removal. Therefore only cross sections are displayed

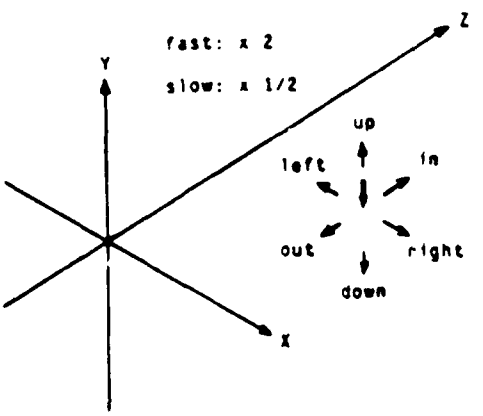


Figure 7: 3D pointer controled by voice commands

instead of a Generalised Cylinder while the Generalised Cylinder is being fitted. Besides, hidden-line removal is not necessary as long as the stereo line drawing is used as there is no ambiguity of 3D structure in the stereo line drawing. Figure 13 shows Generalised Cylinder descriptions generated by the system after all the fitting operations are finished.

Although it might be accurate enough for most purposes (The actual size of the part is 31mm x 33mm x 16mm), the Generalised Cylinder descriptions would have been more accurate if we had measured camera parameters more accurately.

It takes about three or four minutes to fit one Generalised Cylinder with four knots, though the problems of display time and 3D input accuracy had to be solved. The total time required to fit Generalised Cylinders to all the subparts of an object depends on how many subparts it has. For example, if the object is formed by two or three subparts, ten minutes would be enough for building the model of the object. With this user-friendly interface, even a person who is not a computer expert can learn how to build models easily and can built them simply and quickly without errors.

5. Conclusions

The Stereo Modeling System which enables simple, fast model building was described. The key ideas here are the following: (1) build a model of an object instance from a 3D example of the object. (2) learn the class model of the object from examples. This system is easy to learn, i.e., even a person who is not a computer expert can master it in an hour or so and models of 3D objects can be built easily and quickly, e.g., a model of a simple object with two or three parts can be built in 10 minutes without errors.

Although the problem of the accuracy of the modeling was solved by the zoom function and the 3D pointer which can be moved in 3D space within one pixel of the stereo images, this modeling system might not be suited for the highly accurate modeling of complicated objects. However, the simplicity rather than the high accuracy of 3D model descriptions is important in Image Understanding. Even we don't have accurate 3D models of objects in the world in our brains. Although the 3D models which this system produces might not be suited for CAD or Computer Graphics in which a high degree of accuracy is important, they are believed to be accurate enough for most applications in image understanding.

It would be ideal if 3D models were built automatically without any aid. However, it is still difficult to build useful 3D models for Image Understanding fully automatically. In this system, the example of the object is divided into parts according to the required accuracy based on human judgement and geometric features of the object are taught and described efficiently using the knowledge of the object we have. Models can be built from one stereo pair for we know that there is no fault in the hidden part of the object. However, a fully automatic system without such knowledge requires pictures of the object from all different angles. Nevertheless, it is true that in our system, a stereo pair from a good viewpoint is needed and several stereo pairs might be required for a complicated object. It is possible to continue building the model from different angles if camera parameters are known and this may be necessary in future. However, the key idea here is *to build models directly in 3D space using actual examples*, i.e., 3D space need not be stereo images and stereoscope. It is also possible to fit Generalised Cylinders using knots in actual 3D space via a 3D pointer like a robot arm. This type of alternative method might be useful if a good 3D pointer is available.

Some classes of objects have various geometric instances and they are best described by functional features and higher geometric features.[8] On the other hand, there is another class of objects which can best be described by geometric features and is difficult to describe by functional features, e.g., there is no man with *10 legs* and what are the functional features of a man ? Also there is a class of objects which can be described both functionally and geometrically, e.g., tennis balls are always round, matchboxes are rectangular, and they have functions. As the first step towards the powerful modeling system which can handle both functional and geometric features, the geometric modeling system described here is designed to build models of objects which can be described by geometric features. This system and the ACRONYM system do not yet handle functional and higher geometric features. They would become more powerful if, in future, they could handle these features.

References

[1] Brooks, R.A., *Symbolic Reasoning Among 3-D Models and 2-D Images*. Ph.D. Thesis AIM-343, Department of Computer Science, Stanford University, Stanford, California, June 1981.

[2] Baumgart, B.G., *Geometric Modeling For Computer Vision* Ph.D. Thesis AIM-249, Department of Computer Science, Stanford University, Stanford, California, October 1974.

[3] Binford, T.O., *Visual Perception by computer*. Proc. IEEE Conf. on Systems and Control, Miami, December 1971.

[4] Newman, W.N. and Sproull, R.F., *Principles of Interactive Computer Graphics* McGraw-Hill, New York, 1979, pp. 154-158.

[5] Slama, C.C. (ed.), *Manual of Photogrammetry*. American Society of Photogrammetry, Maryland. 1980

[6] Baker, H.H., Binford, T.O., Malik, J., and Meller, J.F., *Progress in Stereo Mapping*. Proc. Image Understanding Workshop, Arlington, Virginia, June . 33.

[7] Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., *Machine Learning*. Tioga Publishing Co., Palo Alto. 1983.

[8] Winston, P.H., Binford, T.O. Katz, B., and Lowry, M.R., *Learning Physical Descriptions from Functional Definitions, Examples , and Precedents*. National Conference on Artificial Intelligence, Washington, D.C., 1983, pp. 433-439.

Appendix F

ASTERIX: STEREO MATCHING

ASTERIX (A Stereo Image system) has been partially implemented both as part of a general vision system in succession of ACRONYM and as part of the vision system for a mobile robot. The vision system has to handle stereo pairs as well as motion sequences and should be able to incorporate depth clues from other sensors. ASTERIX is a general system which can be used for the following:

- pairs of stereo images
- motion-sequences
- generation of image descriptions which can be understood by the high level system

Feature Based Approach to the Correspondence Problem

Both motion and stereo vision have the correspondence problem in common. The correspondence problem in the two domains differs only in the type of the constraints which can be applied to solve the problem. The main task however, selecting features in all frames of a motion sequence or a stereo pair, is the same. That is why the ASTERIX-system is based on a general correspondence algorithm which can be applied to motion sequences as well as to stereo pairs.

Specialized knowledge about disparity constraints is kept in separate program modules (constraint sources), which evaluate e.g. epipolar lines (stereo) or velocity predictions (motion). These separate constraint sources provide also an easy way to feed constraints derived from other kinds of sensors into the system. Thus a change in robot configuration affects only the constraint sources and the rules how to apply them, but not the general concept of the system.

Another advantage of this system structure is that in the case of motion sequences the constraint sources can change according to the knowledge acquired so far. In the first frames of an image sequence only very general heuristic constraints can be used, but as more and more about the scene is known more sophisticated constraints can be applied.

Matching Strategy

By grouping and grading of features the matching process can be guided by a plan to deal with ambiguous feature constellations in an "intelligent" way.

ASTERIX first inspects the feature constellations in both frames and plans a matching sequence which seems to be the most promising for the situation. Three levels of features are available for matching:

1. Simple local features are likely to be abundant and present in both (or all) images. They can be computed in near real time using convolution hardware. But they are likely to be indistinguishable from each other.

2. Larger derived local features, such as lines and their junctions, are less abundant. They also have richer descriptions, which reduces the number of possible matches. Also items like T-junctions, which might be caused by occlusion may be excluded.

3. Higher level features, possibly describing an entire object, are easily matched. But they are likely to be distorted by occlusion, perspective or illumination. E.g. if only three corners of a square are visible the square can not be matched, but the corners can.

The matching strategy takes the abundance of features into account. Consider for example the image of a checker board: All corners in the inner part of the board have look-alike neighbors. Any matching strategy based on optimising a similarity measure within a local neighborhood is bound to come up with ambiguous matches or worse, will force an arbitrary decision and select the best match according to the similarity measure, which might depend purely on the noise in the image. Matching strategies based on a global optimization criterion on the other hand have the tendency in case of ambiguities to force some wrong matches for the sake of a globally better optimisation.

ASTERIX uses a different strategy: The system first scans both frames independently and groups the features into classes of similar features, which might cause ambiguities. In the case of the checker board ASTERIX would see immediately that there are lots of look-alike corners in the inner parts of the board, but that the four corners of the board itself are unique. So the system starts matching by matching one of these unique corners first. Then the neighboring corners are matched by following the edges that connect them, until all parts of the image are matched or no more constraints can be derived. Ambiguities which cannot be solved at this stage of analysis are forwarded to the higher levels of analysis, rather than forcing a decision based on low level clues only.

The way ambiguities are solved in ASTERIX is related to graph matching techniques. Both strategies are feature based, evaluate all possible ambiguities and derive constraints from structural knowledge like neighborhood relations. The difference is that graph matching strategies compare features between the frames to be matched whereas ASTERIX groups the features and compares them mainly within the frames. Graph matching is very mechanical whereas the grouping generates a plan for a data-driven matching sequence.

The planning of the matching sequence in ASTERIX can be compared to the way a child tries to solve a jigsaw puzzle. Usually the child will start with pieces that catch the eye because they are strikingly different from all other pieces in either shape or color, and the more advanced child will organize his pieces and sort them according to color and shape.

Feature Extraction and Image Description

Points of interest and edge elements are the basic features. The edge appearance model is used for edge detection. It takes the point spread function used for image generation into account and is capable of locating an edge with up to 1/8 pixel accuracy. Therefore good edges can be located with pixel accuracy in an image reduced by a factor 8, thus resulting in a speed gain by a factor of 64. The rich edgel description, i.e. quality, position, direction, left and right grey levels, results in relatively unique stereo matches even at the edge l level.

Feature generation and image description proceed in 7 stages:

1) A set of gradient operators is applied to the whole image. Pixels above a threshold are marked as candidates for the edge or point of interest operator.

2) At the first unprocessed location the edge operator is applied (without its early stage, which is the gradient operator). If successful the area is marked processed and predictions for the next location are generated.

3) The edge operator is applied at the predicted locations, thus tracking the edge. The quality threshold is set very low in order to negotiate corners, thereby avoiding the search for a new starting point.

4) The resulting string of edge 1's are broken into pieces with sufficient quality. These are further segmented at points of change of curvature, left or right grey level.

5) Second level features - line descriptors - are computed.

6) The ends of lines are extrapolated in accordance with the edge model. Where ends of lines cross, third level features junctions are generated. Prerequisite for a junction is that participating lines intersect at one point and grey levels match locally. Thus partial junctions (one grey level match missing) classes of junctions are:

   - L-shaped corner
   - continuation (L with 180 degree angle)
   - Y-junction
   - T-junction
   - arrow junction
   - star (4 lines meeting, various subclasses)
   - partial junction (inner L, outer L, suspected T, etc.)

Line descriptors are updated with the exact location of their endpoints or merged in the case of "continuation". The network of lines and junctions and associated labels constitutes the image description forwarded to the high level reasoning system.

7) The point of interest operator is applied to the remaining candidate locations. It labels the points of interest as dark objects on a light background or vice versa and computes also the orientation of the corners and a measure of quality.

Grouping, Grading and Matching

The features are grouped into equivalence classes. Matching takes place between classes of features rather than between features themselves. The first step of matching is grading of the classes according to abundance and prominence of features. The matching sequence is determined by the grades of the classes. The classes with only few but very prominent features (high contrast) are matched first to get reliable and unique feature matches. From these initial matches new constraints are derived and propagated into the more ambiguous classes. The constraint propagation follows the edges which connect the corners and junctions.

The grouping and grading of features does not necessarily reduce the computation time for the matching. The comparison of features within the frames takes about the same time as the comparison of features between the frames, but the grouping process saves the results of the comparisons in a more useful way, so that a plan can be derived to solve ambiguous constellations.

Implementation and Results

TV cameras are connected to a SUN workstation enhanced by image processing boards. The image processing hardware is capable of frame rate pixel operations and does subsampling, averaging and gradients. Edge operator, edge follower and interest operator are implemented in C, all other parts of the system are running in SLISP, a flavor of COMMON LISP.

The feature extraction and the grouping and grading of features have been tested real-world scenes (machine parts, telephone, country road) with quite satisfactory result Preliminary matching results were obtained for the country road pictures.

Appendix G

ACTIVE OPTICAL RANGE SENSING FOR ROBOTS

I. Introduction

Active optical range sensors can have useful applications with mobile and fixed robots. Compared to passive sensing, active sensing reduces the complexity of information processing required for perceiving the environment. Immediate concerns, such as the presence of obstacles, may be addressed quickly without the need for advanced image understanding techniques.

This report will examine the characteristics of two types of ranging sensors. The first are those which measure range by triangulation as in stereo. They measure the angles of the illumination source and its image. Some are known as structured light systems because they cast a pattern of light on the scene and sense the image positions of the light pattern. The second class is LIDAR, which measures time of flight of a light beam either by measuring round trip time directly or by measuring phase angle of a reflected, modulated beam.

II. Sources

Sources fall into two general classes, incoherent and coherent (usually lasers). Incoherent sources offer high power at low cost in the form of arc lamps and flash tubes. Flash tubes measuring a few inches in length can produce pulses of several kilojoules at up to 50Incoherent sources may only be used with triangulation ranging since powerful source types cannot be temporally modulated with short enough wavelengths to be of interest in robotic situations. Another difficulty of incoherent radiators is fundamental, they cannot be easily focussed into a small beam of low divergence. This leads to poor depth of focus with structured light patterns. When operating outdoors, incoherent sources can have difficulty competing with sunlight. By contrast, laser signals can be discriminated from sunlight with the use of dielectric interference filters or by modulation of the source.

Laser sources offer much as radiators in range sensing systems. They may be easily focussed into small beams of low divergence or projected in patterns with very large depth of focus. Further, they may be modulated at wavelengths suitable for robotic range measurement. In fact, it is feasible, though expensive, to measure absolute range directly to sub-micron accuracy over several meters using advanced laser systems.

The advent of the stabilized laser source has made practical the treatment of the laser in the manner of a radio or radar signal, rather than as just a directable heat source. Using doppler techniques, velocity or surface vibration may be sensed.

From the point of view of radiator power available, CO2 lasers are very attractive. They may be stabilized effectively, and can radiate optical power into the kilowatt range continuously. With heterodyne detectors, CO2 laser signals may be detected at the photon noise limit at very low levels of detector illumination. Because of the relatively long wavelength (10 microns) the photon statistics are better than those of visible lasers for comparable power.

Disadvantages of CO2 laser systems include cost of the laser, high cost of 10 micron optics, high cost of detectors (which normally need to be cryogenically cooled) and the difficulty of alignment which comes with heterodyne photodetection.

For very high bandwidth environment sensing, a CO2 based system would be difficult to surpass, as far as performance is concerned. Small sealed waveguide CO2 lasers and electronically cooled detectors will make advanced long range vehicular systems feasible.

An advantage of CO2 systems is that for a moderate power level, CO2 laser radiation is less hazardous to personnel than laser beams in the visible spectrum, as it will not penetrate the cornea to burn the retina, as visible laser radiation will. As far as safety is concerned, CO2 radiation has the disadvantage of being invisible. This might be an advantage in military applications. Also, the high power of CO2 lasers increases risks of intense exposure.

Helium neon lasers are available as inexpensive radiators at moderate power. Very cheap lasers are available with power in the milliwatt range. With careful attention to signal detection, this can be enough for practical use in robotic environments. Larger units are available with power to hundreds of milliwatts. HeNe lasers may be stabilized and used in doppler systems. It is possible to stabilize a HeNe laser running two cavity modes which provides effective modulation in the GHz range.

Solid state lasers are available commercially with powers in the tens of milliwatts which may be directly modulated at wavelengths useful in robotic applications. Advantages for these units over gas lasers are their small size and ruggedness. One disadvantage can be the need to use beam collection and concentration optics.

YAG lasers can be used in systems measuring time of flight directly by measuring the transit time of pico-second pulses.

Other lasers which might be suitable for robotics applications include Helium Cadmium lasers, which are similar to Helium Neon lasers while offering more power at higher cost.

III. Sensor Systems

The detection system is as important as the source in an active optical ranging system. Noise characteristics have a strong impact on the operating range and radiator power needed. The main types of detectors in use fall into the classes of solid state photodetectors, photo-multipliers, and heterodyne systems.

The criterion of merit is whether the detector is photon-noise-limited at the typical received power with which the system will operate. A system will be photon-noise-limited if the shot noise due to the photon flux of the received signal is the dominant noise in the detection system (over thermal noise, etc..).

It is possible to approach the quantum noise limit at reasonable power levels using either photo-multipliers or heterodyne photodetection. Solid state photosenssors are typically limited by thermal noise.

Solid state sensors are available as array sensors (e.g. CCD television cameras) and as analog position-sensing devices, where several cathodes share the same photodiode and

the position of the beam falling on the sensor is inferred by the ratios of photo currents among cathodes.

Area sensors are limited in bandwidth by their scan frequencies to the audio range. Position-sensing photodiodes are bandwidth limited by their large capacitances to the low megahertz range.

Single photodiodes and modern photomultipliers using micro-channel techniques are available with bandwidths to several GHz, making them practical in systems based on phase-detection of a modulated light source.

Heterodyne photodetection systems are capable of operating at the quantum noise limit with comparatively low detector fluxes. Their disadvantages include the cost of generating optical local oscillator and the necessary alignment of the signal and local oscillator beams on the face of the photo-detector. Heterodyne photodetection is practical for quantum-noise-limited detection of weak CO2 laser signals. Photomultipliers do not respond to CO2 radiation.

IV. Sensing System Strategies

In order to be of use in range determination, the emitted radiation must be patterned either temporally, as in LIDAR systems, or spatially as in structured light systems. We will first discuss structured light systems.

One way of classifying structured light systems is according to the style or patterning of the radiation. The most commonly used patterning method has been "light striping", in which a planar sheet of light is scanned through an angle over time. With an image derived from an imaging sensor, it is then straightforward to determine range by trianulation. The imaging sensor may be an ordinary video camera attached to a frame buffer. Solid state area sensors are often used because of their inherent geometric accuracy. A limitation for a system of this sort stems from the need to digitize a large number of images to capture the sweep of the light sheet through enough angles. For example, if a horizontal resolution of 100 is desired, then 100 frames of video will be required to obtain a dense depth map. This sort of measurement strategy has been used in commercial robotic range sensing systems.

A variant of of this scheme involves basically the same strategy restricted to two dimensions. In this case, the radiator emits a beam rather than a sheet of light which is swept through an angle over time. Here a linear sensor is needed. For this, position sensing photodiodes may be used [Lemarquand]. These sensors allow for relatively simple analog electronic signal processing. For 3-d sensing, this 2-d system may be scanned through another angle. Yet another variation here involves the use of a photo-multiplier tube in conjunction with a rotating mirror as the detector. This scheme was used by [Pipitone 83].

Other methods possible for structured light ranging include "gray-scale" structured light, and "bar-pattern-sequence" structured light. In gray-scale structured light one radiates several patterns as a continuous function of the emisssion angle. At each location on an imaging sensor, it is possible to infer the emission angle of the radiation falling on the corresponding surface point by observing the detected intensity for each of the several radiated patterns. An advantage of a system of this type is that dense depth data could be derived from a small number of emitted patterns, perhaps three. It would be feasible to

implement the signal processing portion of this system using a conventional video stream image processor.

Bar-pattern-sequence structure light is another strategy. Here, a sequence of bar patterns encodes the emitted radiation angle. If binary encoding of the emission angle were used, it should be possible to achieve angular resolution of 256 with a sequence of 8 emitted patterns. Alternative coding, e.g. grey coding, would provide better behavior at pattern boundaries.

LIDAR SYSTEMS

The second major classification of active optical ranging systems encompasses those systems which employ time modulation of the radiation as the pattern which is used to discriminate range. Such systems measure range by inferring the transit time from the radiator to the object and back to the sensor. The transit time is either measured directly as in time-of-flight systems, or indirectly by measuring the phase shift in the light beam of known wavelength as it travels from the radiator to the target and returns.

Typical modulation frequencies for such systems range from the tens of megahertz to thousands of megaherts, with a trade-off between range of unambiguous measurement and range resolution. That is, phase measurements imply distance measurements modulo the wavelength. Longer wavelengths enable a greater range of unambiguous measurement and less range resolution. Because of bandwidth requirements, single photodiode detectors or photo-multiplier tubes are usually used. Small PM tubes using microchannel plates are available with bandwidths into the GHz range. Such sensors are attractive as they allow detector performance which approaches the quantum noise limit, for low signal power levels.

In phase detection of a modulated source, there is a practical limit on the resolution of phase measurements made, typically on the order of one part in 10,000. In order to achieve high range resolution with a large measurement range, one may employ a hierarchy of modulation wavelengths. Measurement at longer wavelengths is used to determine which "cycle" of a shorter wavelength signal is being observed. Then the shorter wavelength measurement may be used to obtain increased resolution.

Direct time of flight measurement is possible using short pulses available from YAG lasers. The advantage in cost and complexity over continuous wave lasers with phase-detection of modulated sources is not clear.

An interesting system has been patented by ITEK which employs the phase angle measurement of laser radiation. ITEK exploits consistency requirements of observations at a set of positions of the target as measured by several beams separated in angle. This system may be seen as a sort of hybrid between stereo type systems and phase-measurement systems. Optical frequency phase angle measurements, interferometry, have been used for displacement measurements as in the Hewlett-Packard interferometers.

V. Far-field comparison of sensing strategies

One point of comparison for active optical ranging systems is the asymptotic behavior of the system resolution at long distances. We will discuss this behavior for laser systems.

Asymptotics of current SNR

All systems have a dependence of resolution, or range noise, on the current signal-to-noise ratio. If laser systems are compared, and the received signal is weak enough that thermal noise dominates quantum noise, then a good model of the current SNR is that it is inversely proportional to the square of the range. The assumptions here are that the detector sees the entire spot of the beam on the target, and that the intensity of radiation arriving at the detector optics obeys the inverse square law.

Relation between phase resolution and current SNR

In phase-measurement systems, the relation between current SNR and measured phase resolution is important.

A phase measurement scheme which is particularly easy to analyze is one where the timing of zero crossings in the sinusoidal signal is measured. Such a detector can theoretically be made to have good noise performance by using a narrow band pass filter before the zero crossing detector. It is easily seen that the size of the fluctuation in the position of a zero crossing will be proportional to the fluctuation in the current divided by the slope of the signal at the crossing. Since the slope of the signal at the crossing is proportional to the amplitude of the signal current, it is clear that the fluctuation of the position of the zero crossing and hence the noise in the phase measurement will be inversely proportional to the current SNR.

In a phase detection system, the range resolution is proportional to the phase resolution, since range is proportional to phase. Because the phase resolution is inversely proportional to current SNR, which is inversely propotional to range squared, we see that range resolution is proportional to range squared.

Relation between range resolution and angle resolution

To ascertain the long distance asymptotics of structured light, i.e. triangulation ranging systems, we need to determine the relationship between angular resolution and range resolution. Examining the case where the range is much larger than the baseline and the triangle is Isoceles, one may approximate the baseline as the range multiplied by the small apex angle. Since angle increments measured are proportional to the apex angle (the factor is .5) we see that the measured angle is proportional to the baseline divided by the range. Differentiation shows that the increment in range is proportional to the increment in angle multiplied by the range squared and divided by the baseline. Thus for fixed angular resolution, the range resolution is inversely proportional to the square of the range for stereo type systems.

Relation between angular resolution and Current SNR

The asymptotics of triangulation optical ranging systems also depend on the relation between the resolution of angular measurements and the current SNR. An easily analyzed detector for angles may be arranged by imaging the distant spot onto a position-sensing photodiode. Here the imaged spot position, and hence the angle, is proportional to the difference between two photo currents divided by their sum. If the noise currents are small and comparable, then the angular resolution will be inversely proportional to current SNR, thus inversely proportional to the square of the range.

Since we've seen that for fixed angular resolution, the range resolution of a triangulation system is inversely proportional to the square of the range, and since the angular resolution is also inversely proportional to the square of the range, we see that the range resolution of this position-sensing photodiode is actually inversely proportional to the fourth power of the range.

Asymptotics Summary

For systems having comparable near-range performance there is a disadvantage for triangulation systems due to their additional inverse square range factor in their far-field resolution.

Range noise criterion comparison

One way to compare proposed or realized active optical ranging systems is by formulating a quantitative performance criterion. This criterion should factor out differences in the experimental conditions of the measurements such as the power of the laser, the operating range, and the measurement time, since it is unlikely that these parameters will be the same.

A measure of interest in a ranging system is the range noise. This is often measured as the rms fluctuation in range.

Since averaging uncorrelated measurements will reduce the noise by a factor of the inverse square root of the number of measurements, we should multiply the rms range noise by the square root of the measurement period to factor out differences in measurement times.

Since the range resolution is inversely proportional to current SNR, we should multiply by the effective detector illumination to penalize those systems using stronger lasers, or else working closer in. Thus we should also multiply the rms range fluctuation by the laser power and divide by the range squared.

Thus our criterion of comparison is the rms range noise, per root hz. of measurement bandwidth, times the detector illumination factor.

We will compare three systmes using this criterion. These will be the systems of [Lemarquand 83], [Pipitone 83], and a hypothetical system extrapolated from that of [Duda 79].

Lemarquand's system is a triangulation system which uses a linear position-sensing photodiode as the angular sensor. The laser power is 5 mw., the rms range resolution is .5 mm. at .5 m, and the mesurement rate is 64k measurements per second. Our criterion of comparison is then: $3.9\ 10^{-8}$

rms meters per root hz. watts per / meter squared.

Pipitone's system is also a triangulation system which uses a photomultiplier tube and a rotating scanning mirror as the angular sensor. This system also employs a laser of approximately .5 mw., and has a resolution of .25 inch rms in 96 inches. The measurement rate is 500 Hz. Our criterion of performance is then: $2.42\ 10^{-7}$

rms meters per root hz. watts per / meter squared.

The hypothetical system we will examine is based upon Duda et al's system but differs in having the modulation frequency increased by a factor of 100 from 9 Mhz to 900 Mhz. Such a modulation frequency is within the bandwidth of modern microchannel plate photomultiplier tubes. It was pointed out above that the ambiguity arising from having the wavelength shorter than the range can be resolved by also operating with a longer wavelength. The system of Duda et. al. had a range resolution of 1 cm, so we expect to achieve a resolution of .1 mm by scaling wavelength. The operating range was about 2 meters, and the measurement time was approximately .3 sec. the system employed a laser of approximately 5 mw power. Our criterion of system performance is then: 1.39 10^{-8}

rms meters per root hz. watts per meter squared.

VII. Conclusions

We have discussed the basic types of active optical range sensors, their radiators, sensor systems, and sensing strageies. Laser radiators are seen to have an advantage in daylight applications, and as radiators in phase-detection systems.

Triangulation systems are seen to have a far-field disadvantage when compared to modulation/phase-detection systems having similar near-field performance.

Three different systems have been compared using a quantitative measure of performance which factors out differences in measurement time, laser power, and operating range. The three systems are seen to have roughly comparable near-field performance.

Because of the high power available, CO2 laser-based systems could provide very high performance at high cost. Helium Neon laser-based systems could provide good performance at more moderate cost.

References

[Duda 79] Duda, R., Nitzan, D., and Barett, P., "Use of Range and Reflectance Data to Find Planar Surface Regions", IEEE Trans Pattern Analysis and Machine Intell. 1 pp 259-271, 1979.

[Lemarquand 83] D. Lemarquand, "Reconnaissance et Positionnement d'Objets dans le 3 Dimensiones", Doctoral Thesis, Universite de Paris-Sud, 1983.

[Pipitone 83] Pipitone, F., and Marshall, T., "A wide field Scanning triangulation rangefinder for machine vision", International Journal of Robotics Research, 2, 1983.

Appendix H

EFFICIENT GAUSSIAN FILTERING VIA BOXCAR CONVOLUTION

William M. Wells III

I. INTRODUCTION

This report examines current approaches to the economical computation of Gaussian Convolutions, and presents an algorithm for this computation which is based on the succesive convolution of the input image with simple "boxcar" functions.

Convolution of image data by Gaussian kernels is a popular operation in image understanding. Burt demonstrates the utility of this primitive in image data compression.

The approach presented here has results equivalent to one of Burt's methods (under certain conditions). Besides being very efficient, it is also appropriate for implementation in hardware.

II. GAUSSIAN CONVOLUTION IN IMAGE UNDERSTANDING

Convolution by a Gaussian function amounts to low-pass filtering or blurring of the input image. This operation might seem contrary to the aim of image understanding. In their work on edge detection, Marr and [illegible] propose that meaningful intensity changes in an image occur at various spatial scales, and that they are best located by the peaks in the first spatial derivative, or equivalently, by zeroes in the second spatial derivative of the low-pass filtered image. Their desire for a radially-symmetric second derivative operator led them to choose the Laplacian applied to the Gaussian function of radius as their filter or operator. The size (standard deviation) of the Gaussian may

be adjusted for the desired spatial scale. Edges in the image at the chosen scale are then marked where the result of this filtering crosses through zero. These zero crossings form closed contours. Some kind of thresholding is usually used, for instance, to suppress edge marking if the magnitude of the gradient at a zero crossing is below a constant value.

The Laplacian applied to the Gaussian is a "mexican hat", or "center-surround" type operator. For reasons of efficiency, owing to the separability of Gaussians (more about this in section III), this operator is often approximated as the Difference-Of-Gaussians of two scales (DOG).

There are several reasons for the choice of the Gaussian as the smoothing function to use in a scheme of this sort. Desirable qualities of a smoothing function include effective low pass filtering and spatial localization. The Gaussian is unique in simultaneously optimizing both in space and frequency [Marr 80]. Marr and Hildreth propose that a suitable representation for visual imagery is a set of zero crossing data of the Laplacian of Gaussian operator at heirarchically arrayed scales.

Burt [Burt 81] proposes a representation which is a heirarchy of DOG images of exponentially-related scales for use in a data compression scheme. Crowley [Crowley 84] employed such a representation in his research in image understanding. With suitable boundary conditions, this type of representation is easily seen to be complete, as the original image may be reconstructed exactly by adding up the components

of the heirarchy.

Canny, in his recent work on edge detection [Canny 83], computes optimal edge-operators for certain conditions which are approximated as derivatives of Gaussians.

III. CURRENT APPROACHES TO COMPUTING GAUSSIAN CONVOLUTIONS

One commonly exploited feature of the Gaussian function of radius is its separability into a function of x multiplied (or convolved) by a function of y. For an operator limited to an N by N non-zero region of support, this reduces the cost of the convolution from N^2 to 2N multiplies and adds.

Another approach used in approximating the Gaussian distribution exploits the Central Limit Theorem, in that repeated convolutions by simple distributions tend to approximate the Gaussian, or Normal distribution [Feller 57]. Canny [Canny 83] exploits this to improve his recursive filter approximation of Gaussian convolution. The successive convolution by box car method we present below is also based on this approach.

A simple example of such a scheme is to successively convolve by the one dimensional kernel (1, 1). This is equivalent to convolving once by a binomial coefficient distribution. These distributions have long been approximated with Gaussian distributions by statisticians [Feller 57].

This binomial distribution scheme is appealing in some contexts as no

multiplications are required in the computation. It isn't the most attractive, as will be shown below, since the standard deviation of the resulting Gaussian approximation grows only as the square root of the number of successive convolutions of the simple kernel.

THE METHODS OF BURT and CROWLEY

Burt and Crowley develop similar methods for computing a heirarchy of Gaussian-like convolutions at exponentially related scales. Both employ a "generating kernel" with a small number of non-zero elements, typically 5 by 5 for 2-D applications. Burt [Burt 80] describes two methods "Hierarchical Discrete Correlation" and "Reduced Hierarchical Discrete Correlation". Burt [Burt 81] expands upon the second method as applied to efficient signal encoding.

Burt's two methods may be described recursively as follows:

1.

- The first member of the heirarchy is the input image.
- The n+1'st member of the heirarchy is computed by convolving the n'th member of the heirarchy by the generating kernel after it has been "expanded" by a factor of 2.

2.

- The first member of the heirarchy is the input image.
- The n+1'st member of the heirarchy is computed by "reducing"

the n'th member of the heirarchy by a factor of s and then convolving by the generating kernel.

In the case where s = 2, the expansion consists of inserting a zero between each element of the kernel. for example, transforming the kernel (1, 2, 1) into (1, 0, 2, 0, 1). When s = 2, the reduction amounts to throwing out every other element of the distribution, for example, the distribution (1, 2, 3, 4) could be transformed into (1, 3) or (2, 4).

Burt examines the case where s = 2. Crowley's method amounts to taking s to be the square root of two and combining the two above methods in an alternating fashion.

These methods produce a set of results closely approximating the convolution of the input image with Gaussian kernels having exponentially related scales. A heirarchical DOG representation may be had by subtracting neighboring levels in this Gaussian heirarchy.

CANNY'S APPROACHES

Canny [Canny 83] discusses several interesting methods for computing approximations of Gaussian convolution. One of these is based on recursive filtering, another is an adaptation of Strasser's algorithm for fast multiplication.

The recursive filtering method uses a two-pole, two-zero (four term) recursive filter whose response is a damped exponential cosine which approximates the Gaussian. Filters of this sort have causal, or

one-sided, responses. To overcome this, he applies the filter to the input twice, once in each direction (for each dimension) and adds the results, obtaining a symmetrical response. In order to improve the approximation to the Gaussian he usually repeats the procedure (taking advantage of the Central Limit Theorem, as mentioned above).

A major attraction of this method is that the computational complexity is independent of the kernel size. For each pass in two dimensions, six multiplications and additions are required per pixel. His typical usage of two passes leads to 12 multiplications and additions, for two dimensions, independent of kernel size.

Curry also examines Strassens methods for fast multiplication, with convolution replacing multiplication. An advantage of this method is that it will work for general kernels, not just those which are separable (for example). He finds speedup for some kernel sizes by a factor of 5 or so over brute-force type methods.

IV. REPEATED CONVOLUTION BY BOXCARS

EXPLANATION OF METHOD

The method we are presenting for efficient Gaussian-like convolution consists of repeatedly convolving the input by "boxcar" functions. The fact that the result approximates convolution by the Gaussian function may be seen as a consequence of the Central Limit Theorem.

An attractive feature of boxcar functions is that their convolution may be computed with only two adds per pixel (no multiplies) per dimension, independent of their width.

In one dimension a boxcar function may be defined as:

$$b_N(k) = \begin{cases} 1 & 0 \le k < N \\ 0 & \text{otherwise} \end{cases}$$

where N is the width of the boxcar function.

The proposed method of boxcar convolution is described by:

(1) $$y(k+1) = y(k) - x(k) + x(k+N)$$

where $x(k)$ is the input distribution, $y(k)$ is the output distribution, and N is the length of the boxcar function.

Applying the Z transformation to (1) and using the shifting property of the Z transformation yields:

$$zY(z) = Y(z) + X(z) - z^N X(z)$$

or

$$Y(z) = \frac{z^N - 1}{z - 1} X(z) = \sum_{n=0}^{N-1} z^n X(z)$$

Hence

$$B_N(z) = \sum_{n=0}^{N-1} z^n$$

represents the transfer function of this operation. It is also the Z transform of the boxcar function.

Thus the proposed computation is equivalent to convolution by a boxcar function. This computation has been called "updating" by Binford.

This is easily demonstrated by examining

$$H_{m-1}^{M}(z) = \left(1 + z^{-2^{m-1}}\right)^{M}$$

(from 2), using the binomial theorem,

$$H_{m-1}^{M}(z) = \sum_{j} \binom{M}{j} z^{-(2^{m-1} j)}$$

and taking the inverse Z transform.

Note that in (4) h_{m-1}^{M} refers to h_{m-1} convolved M times, not raised to the M power as with H_{m-1}^{M}.

From (3) we may write

$$B_{2^m}^{M}(z) = H_{m-1}^{M}(z)\, B_{2^{m-1}}^{M}(z)$$

Taking the inverse Z transform and using the convolution summation property of the Z transform results in:

$$b_{2^m}^{M}(k) = \sum_{i=0}^{\infty} h_{m-1}^{M}(i)\, b_{2^{m-1}}^{M}(k-i)$$

Substituting (4) leads to

$$b_{2^m}^{M}(k) = \sum_{i=0}^{\infty} \sum_{j} \binom{M}{j} \delta(i - j 2^{m-1})\, b_{2^{m-1}}^{M}(k-i)$$

or

$$b_{2^m}^{M}(k) = \sum_{j} \binom{M}{j} b_{2^{m-1}}^{M}(k - j 2^{m-1})$$

With the identification of g_m as $b_{2^m}^{M}$ this may be written as follows.

$$g_0(k) = b_1^{M} = \delta(k)$$

$$g_m(k) = \sum_{j} \binom{M}{j} g_{m-1}(k - j 2^{m-1})$$

This recurrence relation describes the family of kernels whose convolutions are equivalent to M successive convolutions by a boxcar of width 2^m.

If the free parameter is chosen as follows:

(5) $$a = \frac{3}{8}$$

then

$$w(i) = \binom{5}{i+3} \times \frac{1}{16}$$

In Burt's scheme, varying the parameter "a" affects the Gaussian approximation characteristics of the equivalent kernels.

One criteria for choosing "a" is minimization of spurious (in comparison to Gaussian kernels) high frequency responses in the power spectra of the equivalent kernels. Carrying out this minimization reduces the peak value of the spurious lobes to 6 db. below the D.C. response.

With "a" chosen as in (5), the case where Burt's method corresponds to ours, the undesired responses are down by 30 db..

Thus in the above sense our method isn't as optimal as Burt's. In applications with real signals or images, however, the difference is minor.

DISCUSSION OF CONTINUITY

The equivalent convolution kernels which result from M convolutions of boxcars of size n are:

$$b_n^M(k)$$

The continuity of these functions may be studied in the context of their non-discrete analogs, that is, in the limit that the sample spacing goes to zero, with the width being held constant. The continuous analogs of the equivalent kernels are described by the following recurrence relation:

$$b'^{m}_{w}(x) = b'^{1}_{w}(x) \otimes b'^{m-1}_{w}(x)$$

$$b'^{1}_{w}(x) = \begin{cases} 1 & 0 \le x \le w \\ 0 & \text{otherwise} \end{cases}$$

These functions may be convolved analytically, and have the following properties:

function	form	continuity
$b'^{1}_{w}(x)$	piecewise constant	not continuous
$b'^{2}_{w}(x)$	piecewise linear	C^1
$b'^{3}_{w}(x)$	piecewise quadratic	C^2
$b'^{4}_{w}(x)$	piecewise cubic	C^3
	etc. .	

Again, as suggested by the Central Limit Theorem, the degree to which these functions approximate the Gaussian function improves with order.

GOODNESS OF FIT

The following table summarizes a measure of the degree to which the equivalent kernels of the successive convolution by boxcar method

approximate Gaussian functions of the same weight and variance. The quantity shown is the rms difference figured over the non-zero region of the equivalent kernel, divided by the weight of the equivalent kernel.

boxcar length	number of convolutions	rms difference/weight
4	1	.180
4	2	.043
4	3	.021
4	4	.014
8	1	.127
8	2	.026
8	3	.013
8	4	.009
16	1	.090
16	2	.018
16	3	.009
16	4	.006
32	1	.064
32	2	.012
32	3	.006
32	4	.004

VARIANCE AND WEIGHT OF EQUIVALENT KERNELS

The weight of $b_N^1(k)$ is easily seen to be N. Since the equivalent kernels are non-negative

$$\text{weight}\left(b_n^N(k) \right) = N^n$$

The variance of $b_N^1(k)$ may be directly calculated and is:

$$\text{var}\left(b_N^1(k) \right) = \frac{k^2-1}{12}$$

The variance of the convolution of functions of this sort is the sum of the

variances of the individual functions. Thus

$$var\left(b_n^M(k) \right) = M \frac{k^2-1}{12}$$

EXTENSIONS

It is sometimes difficult to obtain desired variance and continuity characteristics when designing an algorithm using our approach. A simple variation which allows more freedom in achieved variance is to convolve by a set of boxcar functions of different sizes.

A characteristic of the boxcar convolution method described above is that features of the input signal or image drift in position. This may be reduced or eliminated by alternating the directions in which the boxcar convolution passes are made (in applications having this flexibility in addressing).

VI. APPLICATION ISSUES

An important concern in choosing an algorithm for Gaussian convolution is whether a hierarchy of results at different scales is desired. If full size (i.e. not reduced) results are desired at only one particular scale, then Burt's method [Burt 81] is more costly, as it requires computation proportional to the log of the size of the kernel edge, whereas Canny's recursive filter approach [Canny 83] and the successive boxcar method we present here have complexity which is independent of the kernel size.

Which is most economical depends on the application. Canny's

recursive filter method [Canny 83], as he implemented it, requires 24 multiplies and adds (48 operations) per pixel for two dimensions. The method we present, as usually used in four passes, requires 16 adds per pixel in two dimensions. On the basis of number of operations, our approach would seem to be more economical. The method we present does require periodic re-scaling to prevent overflow when faced with limited precision integers, large kernel sizes, and low spatial frequencies in the input data. This can be economically handled in most situations with a shift instruction, or a fixed shift in special hardware. So 16 - 24 operations per pixel is a more realistic estimate for a general purpose computer.

In applications where multiplication is substantially more expensive than addition, the method we present would have some advantage. This applies with some computers (eg. Motorola 68000). Adders have less hardware complexity than multipliers in special high-speed processor implementations.

An issue to consider, in addition to number and type of operations, is the addressing requirements (locality and sequencing). Canny's method requires access to a small set of nearby pixels and previous results, independent of the kernel size. Address sequencing in two directions (2D dimension) is required.

Our method accesses data separated by the size of the boxcar, which is comparable in size to the kernel, in each dimension. Our method is applicable in situations with simple sequential addressing. In such a situation storage would be needed for n pixels and n rows for a boxcar

of size n. In situations where more flexibility in addressing is available, the storage requirement can be reduced to 2n pixels. An example of this sort is a processor where the addressing sequence can be changed from row major to column major. Here, the convolution may be carried out in one dimension with one mode of addressing, then in the other dimension with the other mode of addressing.

In contexts where a full heirarchy of results is desired, any of the methods discussed may be considered, since all (except Canny's adaptation of Strassen's algorithem) have comparable complexity to generate the next member of the heirarchy. Burt's methods [Burt 80] [Burt 83] are lent to powers of two in the scale heirarchy. Crowley's method [Crowley 84] is designed for the square root of two. The method presented here, as well as Canny's, could generate heirarchies with nearly arbitrary collections of scales.

VII. IMPLEMENTATION

We have implemented Gaussian convolution using successive convolution by boxcars in a straight-forward fashion on a VAX 11/780 in the C language. Approximately 60 seconds are required for a four pass implementation on 512 by 512 by 8 bit images.

REFRENCES

[Burt 80]
Burt, P. J., "Fast, Heirarchical Correlations with Gaussian-Like Kernels", TR-860, Computer Science Center, University of Maryland ?

[Burt 83]
Burt, P. J. and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code", IEEE Transactions on Communications, Vol. Com-31, No. 4, April 1983, pp. 532-540

[Canny 83]
Canny J. F., "Finding Edges and Lines in Images", Masters Thesis, M.I.T., M.I.T. A.I. Lab Technical Report No. 720, June 1983

[Crowley 84]
Crowley J. L., and R. M. Stern, "Fast Computation of the Difference of Low-Pass Transform", IEEE Transactions on Pattern Matching And Machine Intelligence, Vol. PAMI-6, No. 2, March 1984, pp. 212-222

[Feller 57]
W. Feller, An Introduction to Probability Theory and Its Applications, John Wiley & Sons, 1957.

[Marr 80]
Marr D. and E. Hildreth, "Theory of Edge Detection", Proc. R. Soc. Lond., B 207, 187-217.

[Marr 82]
Marr D., Vision, W. H. Freeman and Co., 1982

Appendix I

ON IMPLEMENTING ATLAS

This part of the report describes experience with the first attempt at implementing ATLAS. It discusses some of the basic issues in dealing with an assembly system, mechanisms for representing, storing and extracting information concerning constraint generation and task specification. It summarises implementation of a skeleton matcher. and points out some elements of interdependency links in the planning system.

Issues concerning implementation of other planning modules, of an improved geometric modelling system and constraint solver, of incorporating dimensional uncertainty in the planning procedure are not discussed. They form a majority of the proposed task planner. We need a careful examination of these parts before we commit to a performance level of ATLAS.

1. Introduction

This report discusses a preliminary implementation in order to develop an architecture for a new task level planning system called ATLAS (Automatic Task Level Assembly Synthesizer) proposed by Brooks and Lozano-Peres[Brooks 83].

Of the few task level systems proposed prior to ATLAS, most concentrated on illustrating a single component of task planning or presenting the syntax and semantics of a task level system with approaches to its implementation. Taylor[Taylor 76] discussed an approach to synthesize sensor-based AL programs from task level specifications. That approach was significantly extended by Brooks [Brooks 83] to include not only forward propagation of symbolic constraints but backward propagation to impose forcing constraints on planning variables. In the process, appropriate sensing is introduced to alleviate deadlock in unsatisfied constraints. It is this approach that underlies much of ATLAS.

Some of the basic driving ideas in the planning process took shape in the form of mechanisms for extracting, representing and storing information concerning constraint generation and task implementation. They have been discussed in detail in section 2.

The constraint generator, skeleton identification, and task specification have been implemented so far.

This report also tries to identify some elements of the thread representing the interdependency among various hierarchically abstracted planning steps. Such interdependency plays the key role in exploiting mutual constraints among various steps to force a decision on planning variables. The extent of integration in planning steps will result from the scope of propagation of these interdependencies in a single task iteration and across successive planning iterations.

Last but not least, the heart of such a planning system is a constraint solver which in addition to the backtracking mechanism built on top, determines the scope of constraint propagation. Issues relating to this aspect of ATLAS were not experimented with in the current effort but need careful examination before committing to a performance level of ATLAS.

2. Issues at stake

2.1 Assemblies

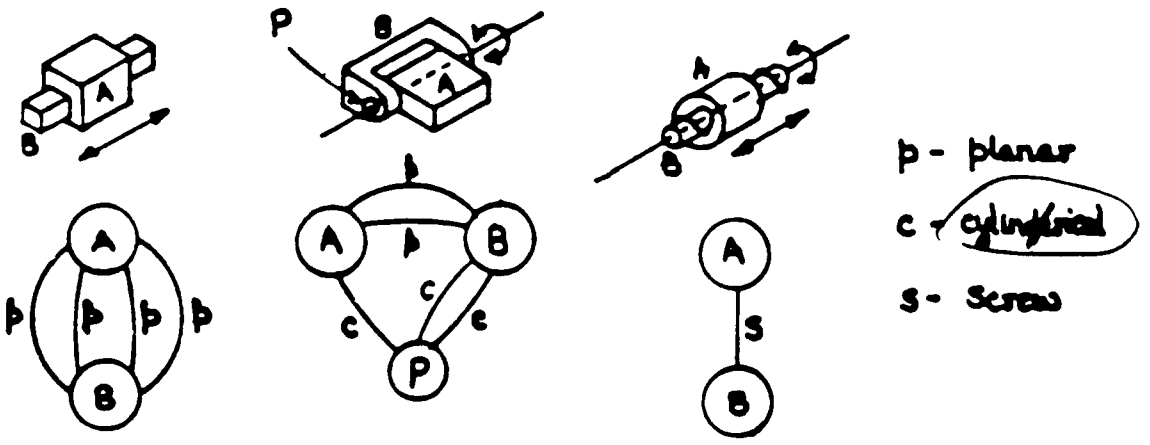
Assemblies consist of objects whose degrees of freedom have been restrained in some manner to some design purpose. Objects are restrained by interaction of features. An assembly is made by a set of operations each of which introduces a new relationship among objects.

Objects interact through their features, namely surfaces, edges and points. Two objects may be related to each other by interaction of single feature elements from each side. We will call such interactions first order interactions. Interactions which involve two feature elements, distinct in at least one of the participating objects, belong to second order and so on. Of the nine possible first order interactions, three are not stable. They are (point,point), (point,edge) and (edge,edge). Others are stable. Stable interactions can be characterised by their connectivity.

By connectivity we mean, the number of parameters that need to be specified to describe an infinitesimally small relative movement of the two objects. Interactions take place over simple geometrical extents of the features involved. At the boundary of features involved in a particular interaction, a small change in relative position may cause the interaction to cease. We will ignore the possibility that small relative movement may end the interaction.

With connectivity defined as above, the first order stable interactions are divided into one of the following categories:

1. Lower pair
   a. Planar interaction - with connectivity 3.



a.Prismatic b.Revolute c. Screw

Some Interactions and their assembly graphs

fig 1

b. Cylindrical interaction - with connectivity 2.

c. Screw interaction - with connectivity 1.

2. Higher pair

a. Point-contact interaction - with connectivity 5.

b. Line-contact interaction - with connectivity 4.

Two objects that share interaction are related. The relationship will at least contain one interaction - first or multiple order. A relationship can be categorized by the residual degree of freedom (RDOF) of either object with respect to the other. In a relationship with first order interaction, the RDOF is the same as connectivity of the interaction. RDOF of a multiple order interaction will be the size of the set of intersection of sets of variables for each participating first order interaction. e.g if the intersection is null, RDOF is 0.

Categorising relationship by residual freedom is a common practice in the study of kinematics. In our case, parts that participate in assembly correspond to links in a kinematic chain and residual freedom between two parts corresponds to the kinematic degree of freedom between two links.

There are two aspects of residual degree of freedom that we will highlight at this stage:

- The position variables of an object that may get associated with any uncertainty are the set of variables present in the residual freedom space.
- Residual freedom of an object changes when it acquires a non-redundant relationship with an object that is introduced by the manipulator.

The reason for deviating from the kinematic division of pairs will become obvious when we analyse the prismatic pair and revolute pair in terms of our types. It will also illustrate the basic difference in recognising the pairs at the level of geometric elements of the two participating objects referred to earlier as features, rather than recognising just the two links involved in the process.

In terms of our division, the prismatic pair is a combination of four planar interactions between the participating objects with one RDOF. The pair could as well be a second order interaction containing two intersecting planar interaction. This configuration, however, will not be physically sustainable.

Similarly, the revolute pair can be achieved in various ways depending on the physical implementation of the pair. One of the implementations has been shown in fig. 1.

Such feature to feature representation of relationships among objects leads to an "Assembly Graph" where objects are represented as nodes and interactions as links. The links in the graph are time tagged.

An Assembly Graph representation of an alternator from Whitney[Whitney 81] has been illustrated in fig 2. The assembly graph has been simplified by not including the detail graph for bearings which theoretically contain higher pair interaction between the races and the bearing elements.

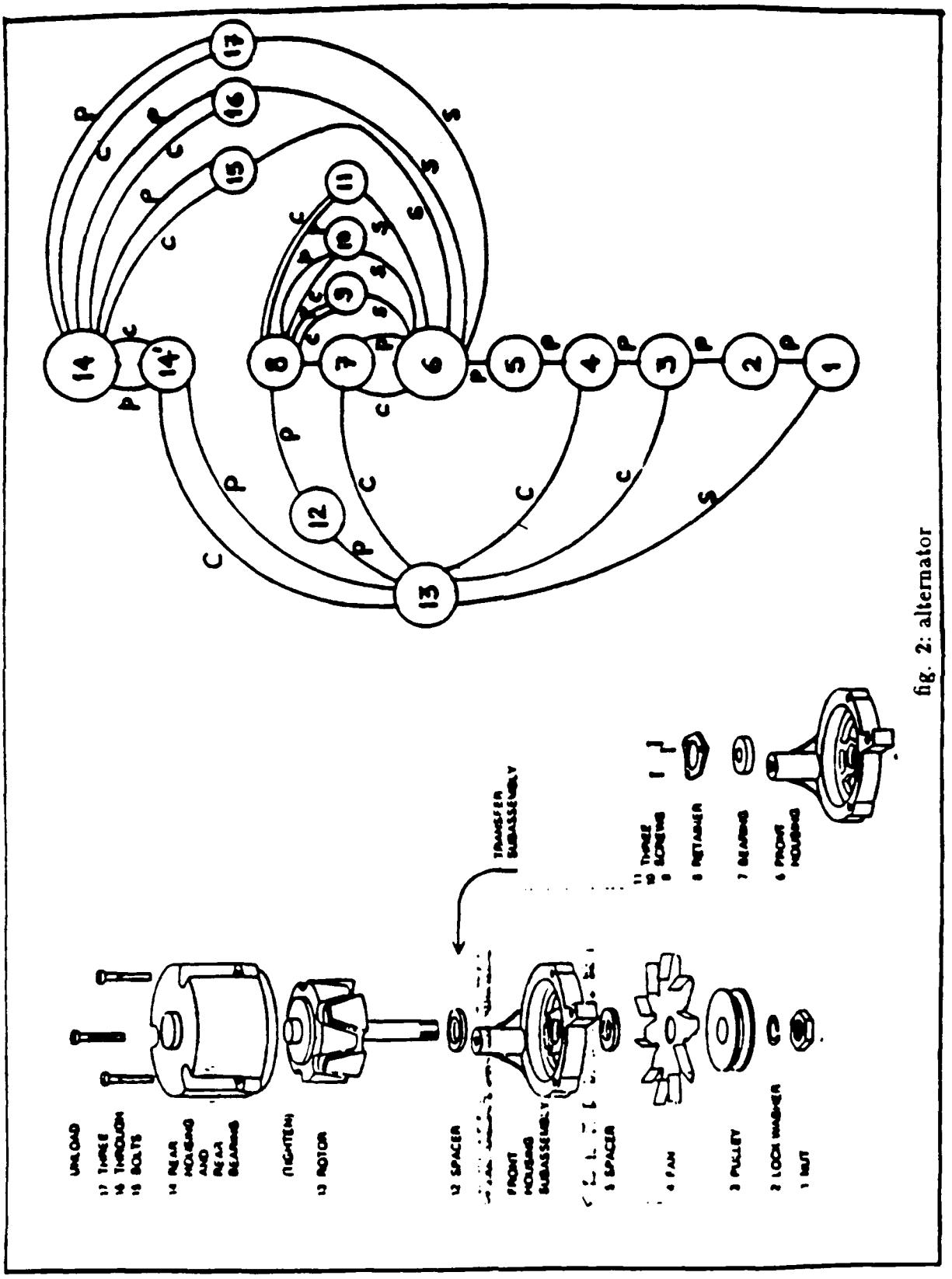


fig. 2: alternator

Assembly Graphs are generated from a specification of the task by the user. An assembly graph facilitates inference of residual freedom of objects at any given time by recognising only those links which have been established at this point. Therefore, the uncertainties in position of an object which are a function of the residual freedom at that instant can be inferred using this graph.

Ambler and Popplestone[Pop 80] recognised some of the relationships described here as ways of specifying spatial relationship and solved the problem of making inference on the spatial position of the object. In the process, they could make an inference of the residual freedom of an object. In a system which aims at simplifying the level of specification of the task, the assembly graph can be generated by compiling the task specification. At the end of specification, the interpreter can return to the user to resolve cases where for the given link constraints, there are multiple solutions for the spatial position of the object or there is no solution because it still has some residual degree of freedom. Issues at this level have not been explored any further.

2.2 Planning Islands

The object of a task planner is to generate robot level program from a given user's specification of the task. The input sequence of the task complete / specifies the sequence of assembly. The planner expands the task into robot specific motions introducing sensory operations to ensure the success of the task under uncertainties.

For example, consider a simple assembly operation where a lid is tightened on top of a box with four bolts; the task can be specified as follows:

Place LID on BOX (against (FACE0 LID) (FACE1 BOX))
Tighten BOLT1 (into (HOLE1 BOX) (HOLE5 LID))
Tighten BOLT2 (into (HOLE2 BOX) (HOLE6 LID))
Tighten BOLT3 (into (HOLE3 BOX) (HOLE7 LID))
Tighten BOLT4 (into (HOLE4 BOX) (HOLE8 LID))

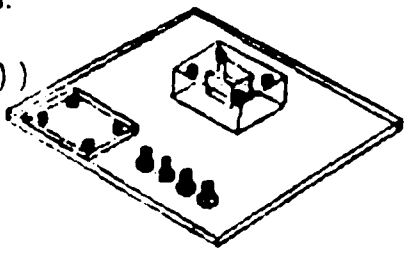


fig. 3

A simplistic approach to generating a robot level program without uncertainties in the manipulator motion or part dimensions will involve making decisions about the following problems:

1. Parts Feeding - considerations here address speed and reliability in acquiring the parts. Feeders may be needed for the *BOLT*$_{i}$, LID and BOX.
2. Layout - This refers to making decisions about the initial locations of the feeders, the location where the assembly will be carried.
3. Fixturing - The planner must determine the fixture which will hold the part for assembly with considerations like reach of certain features, resistance to forces exerted by manipulators, etc.
4. Grasping - The planner should choose the grasp points with considerations of stability in hold, reachability of the features at the initial and goal position etc.

5. Fine Motion - The planner must decide the set of motions that guarantee the success of the task in terms of meeting the goal position specifications, specifically, how the LID will be aligned with the BOX so that the holes are aligned, how the $BOLT_n$ will be inserted etc.

6. Gross Motion - The planner should decide the path of transfer of specific objects without collision.

In reality, a program generated with only these considerations will fail because of the uncertainties in the manipulator motion, uncertainties in the parts dimensions etc. In most cases sensing will solve the problem but this requires that the planner decide what sensing is useful and at what stage it will be helpful. In fact, dealing with uncertainty permeates all of task planning.

Under uncertainty, the robot level program may look like the following:

	1. *Open – Fingers to <width of BOX + epsilon>*
Move	2. *Move to <grasp point of BOX> via <path1>*
BOX	3. *Close – Fingers to <width of BOX – epsilon>*
	4. *Move to <assembly approach location> via <path2>*
	5. *Compliant – Move along <direction> until <contact with Table>*
	6. *Open – Fingers to <width of BOX + epsilon>*
reduce unc.	7. *Vision – Locate <BOX> near <expected position of BOX>*
	8. *Open – Fingers to <width of LID + epsilon>*
Move	9. *Move to <grasp point of LID> via <path3>*
LID	10. *Close – Fingers to <width of LID – epsilon>*
	11. *Move to <approach position of LID> via <path4>*
	12. *Compliant – Move along <vertical> until <contact with BOX>*
	13. *Open – Fingers to <width of LID + epsilon>*
	14. *Open – Fingers to <width of BOLT1.HEAD + epsilon>*
	15. *Move to <grasp point of BOLT1> via <path5>*
Insert	16. *Close – Fingers to <width of BOLT1.HEAD – epsilon>*
BOLT1	17. *Move to <approach position of BOLT1> via <path6>*
	18. *Bolt – In – Hole <BOLT1> <HOLE5 of LID>*
	19. *Open – Fingers to <width of BOLT1.HEAD + epsilon>*

<similarly accomplish insertion of all other Bolts>

Although even this program is not guaranteed to succeed, by introducing sensing the possibility of success has been significantly increased.

The task planner must generate the sequence of operations like placing the BOX first and then realising that the uncertainty in the position of the BOX and the LID will make the task of inserting BOLT1 impossible because of likely misalignment in the BOX and LID. Therefore, introducing a sensing operation before the LID is introduced so that reduced uncertainty in BOX will lead to an improved nominal goal position for the LID thereby, reducing the possible misalignment of the two parts.

Many of the decisions made in generating the above plan will involve numerical calculation of the location of the feeders, the location in the work space where the assembly operation will be carried out, the grasp point, paths for the transfer of parts from one point to another etc.. In addition, for each task, the planner will have to make sure that the constraints necessary for carrying out the present operation has been met. e.g - LID HOLE5 has been aligned with the BOX HOLE1 to sufficient accuracy that the end of the bolt can successfully enter the holes.

Decisions of this nature have been extensively discussed in Brooks[Brooks 82]. The process of planning has been carefully reduced to identifying physical variables, plan and uncertainty variables. Each task is then represented in terms of applicability constraints that need to be satisfied for the task to be successful and propagation constraints that will be propagated further as a result of the accomplishment of this task. The constraints are generated symbolically so that interaction with other such constraints can be solved as a system of constraints. Such a system of constraints allows minimum commitment at any point because a decision is made only when forced to. Later in the process if the plan fails, the decision can be backward-traced by dependency analysis to identify the cause of failure. The planner, can decide whether sensing of the current variable that caused failure can resolve the problem or not. In this process, a backward and forward propagation of constraints is controlled by the planner until a successful plan has been synthesised or the planner returns with a reason for failure.

A system of constraints allows constraints generated across various planning islands to interact if they have variables in common. Thus, in addition to generating constraints properly associated with uncertainties, the planner must also identify links that cause constraints from different planning islands like the applicability constraint of operation 18 (Bolt-In-Hole) to appear in the grasp point decision for operation 10 (grasp for LID). Depending on the distance of the center of rotation of the wrist from HOLE5 of LID, the uncertainty in the location of HOLE5 will be small or large for a given angular uncertainty in locating the wrist.

The next section presents a model for identifying the uncertainties. Later, we will discuss some of the interdependency links that allows constraints from various planning modules to interact.

2.3 Uncertainties

Generating a plan that will succeed in spite of uncertainties is a major part of ATLAS. The exact value of the uncertainties may not be known at the planning time but the planner will need to identify the variables that will be associated with any uncertainty as well as extract information about their possible ranges.

Brooks[Brooks 82] categorised the sources of uncertainties. They are (1) The manipulator (2) the objects to be manipulated and (3) introduction of these objects into the work environment. This categorisation will help in estimating the amount of uncertainty by looking at the error characteristic of the source.

However, these sources of error when perceived from the point of view of knowledge of the world, collapse into two parts namely the positional uncertainty of any object and the dimensional uncertainty in the objects themselves.

At this point, we will illustrate the notion that positional uncertainty is directly related to the residual degree of freedom of the object. In {x,y,s,theta,phi,psi} representation of position of an object, the LID kept on top of the BOX will have {x,y,theta} associated with any uncertainty. More generally speaking, if BOX which is supporting the LID has dimensional uncertainty in the height, then other variables {s, phi,psi} also have associated uncertainty. However, in the absence of any dimensional uncertainty, the variables associated with uncertainty are solely determined by the residual degree of freedom of the object.

Estimating the uncertainty associated with the position of an object which is placed in the workspace will depend on the mechanism of introduction like mechanical feeders or conveyor belts. The residual degree of freedom analysis using the assembly graph of the feeder or equivalent representation of the conveyor can be used to find uncertainty associations. But in order to determine the magnitude, the task planner will have to solely rely on some a-priori information or carry a perturbation kind of analysis on the assembly graph to determine the extents of freedom. This aspect of uncertainty derivation has not yet been explored but it provides a powerful mechanism that can be exploited in a more detailed implementation.

Analysis of the uncertainty in the position of objects due to uncertainty in positioning the manipulator is as follows:

Every time an object is moved in the workspace, the object's position is derived from a known assembly relationship of this object with respect to another object that has already been introduced except in the case of an initial fixture whose position will be decided by the planner. This derived position of the object with respect to a stationary frame is given to the manipulator as the goal position.

At the time the object is ungrasped at its goal position, the position of the object with respect to a stationary frame of reference can be derived either following the kinematic chain of the manipulator or following the object chain in the assembly graph starting from some object whose position with respect to the stationary frame is known and following links leading to objects that will acquire some relationship with this object. The two estimates of position will not usually correspond because of uncertainty in the manipulator chain and uncertainty in the object chain as well, even assuming that there is no significant movement in the object at the time it is ungrasped -a smooth ungrasp operation. Following the object chain and analysing the relationships that this object acquires when being ungrasped, the residual degree of freedom of this object can be ascertained. The object can theoretically be present anywhere in the space of its residual degree of freedom. On the other hand, since the object is a part of the manipulator, the object can be anywhere in the manipulator error ball that the end of the manipulator traces under perturbation of

all of its degrees of freedom with the current nominal position of its links at goal position. With the assumption that the ungrasp operation is smooth, the set of feasible physical positions of the object will be the common subset of the set of feasible positions obtained by the two methods.

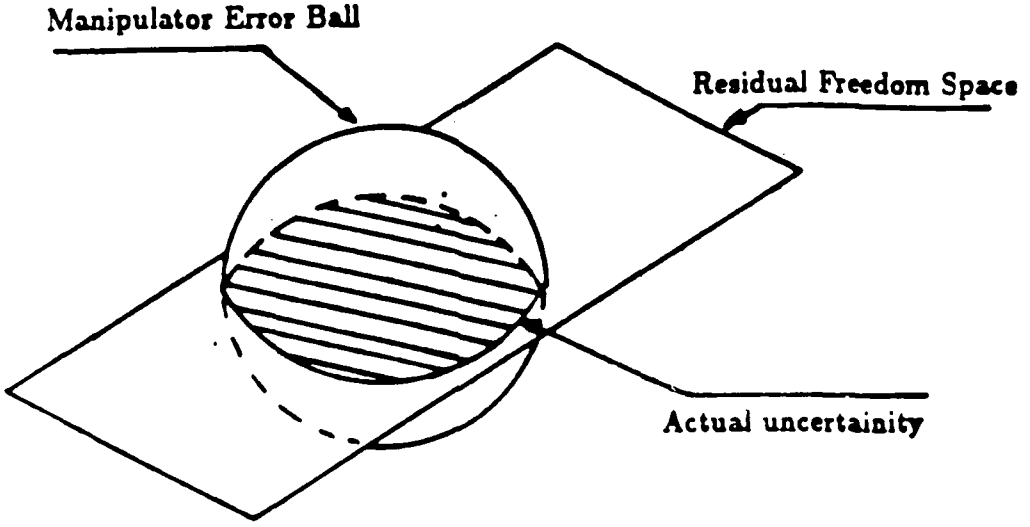


fig. 4

The space of residual degree of freedom of an object is the domain of the residual freedom variables. However, finding the manipulator error ball of the manipulator under perturbation of all of its degrees of freedom for all possible link configurations is virtually ruled out because of the complexity. A simplified model of the error characteristics of the manipulator can be used instead. It is towards this that we will focus our attention.

Let us kinematically divide the manipulator into two portions, namely the gripper that holds the object to be manipulated and the rest of the manipulator. The gripper portion of the manipulator remains integral with the object while the rest of the manipulator can change orientation with respect to the object.

Let $[T_{b,a}]$ refer to the transformation required for a vector specified in frame 'b' to be converted into that of frame 'a'. Then a set of transformations that transforms the object at the manipulator end into the stationary world co-ordinate system can be obtained by following the kinematic chain of the manipulator.

$$[T_{obj,world}] = [T_{link1,world}] * [T_{link2,link1}] \cdots [T_{gripper,link'n'}] * [T_{obj,gripper}]$$

Since the gripper is temporarily rigidly fixed to the object and the first link of the manipulator is fixed with respect to the stationary world, the above set of transformations can be reduced to:

$$[T_{obj,world}] = [T_{link2,world}] * \cdots * [T_{link'i',link'i-1'}] * \cdots * [T_{grip-obj,link'n'}]$$

If these transformations refer to the actual positions, then we can decompose them into the nominal component and uncertainty component by expanding each of the elements of the transformation matrix in Taylor's series in terms of free variables at this joint about nominal values of these variables.

$$i.e [T_{b,a}] = [T^{nom}_{b,a}] + \delta[T_{b,a}]$$

where $\delta[]$ refers to the following expression for a single free variable called 'Sfree'.

$$\begin{aligned}\delta[T_{b,a}] &= \delta Sfree * [T_{b,a}]'_{sfree=nom} + \textit{Higher derivative terms} \\ &= \Delta[T_{b,a}] + \textit{Higher derivative terms}\end{aligned}$$

With these expansions the transformation of the object with respect to the stationary frame following the kinematic chain of the manipulator can be expressed as following:

$$\begin{aligned}[T_{obj,world}] =& [T^{nom}_{link2,world}] * \ldots * [T^{nom}_{link'i',link'i-1'}] * \ldots * [T^{nom}_{grip-obj,link'n'}] \\ &+ \Delta[T_{link2,world}] * .. * [T^{nom}_{link'i',link'i-1'}] * \ldots * [T^{nom}_{grip-obj,link'n'}] \\ &+ ..... \\ &+ [T^{nom}_{link2,world}] * .. * \Delta[T_{link'i',link'i-1'}] * .. * [T^{nom}_{grip-obj,link'n'}] \\ &+ ..... \\ &+ [T^{nom}_{link2,world}] * .. * [T^{nom}_{link'i',link'i-1'}] * .. * \Delta[T_{grip-obj,link'n'}] \\ &+ \textit{higher order terms.}\end{aligned}$$

Therefore, the uncertainty in an object's position involves all the terms of the above expression minus the nominal term which is the first term. Hence, the uncertainty is

$$\begin{aligned}Position =& \Delta[T_{link2,world}] * .. * [T^{nom}_{link'i',link'i-1'}] * .. * [T^{nom}_{grip-obj,link'n'}] \\ &+ ..... \\ &+ [T^{nom}_{link2,world}] * .. * \Delta[T_{link'i',link'i-1'}] * .. * [T^{nom}_{grip-obj,link'n'}] \\ &+ ..... \\ &+ [T^{nom}_{link2,world}] * .. * [T^{nom}_{link'i',link'i-1'}] * .. * \Delta[T_{grip-obj,link'n'}] \\ &+ \textit{higher order terms.}\end{aligned}$$

Using this model of the uncertainties introduced by the manipulator, we will illustrate that for different planning islands different first order terms contribute significantly to the uncertainty of the object. With this assumption, the uncertainty for that island will be modelled to take care only of that term. For some modules, the higher order terms have been lumped together to approximate the uncertainties.

The discussion on links at the level of planning islands contains the details of error modelling.

3. Implementation specific details:

The ATLAS proposal outlined various planning modules or islands that need to be implemented for a full scale implementation of the task planner. These modules are (1) skeleton matcher (2) constraint propagator (3) fine motion planner (4) gross motion planner (5) grasp planner.

Of these, the skeleton matcher has been developed successfully with a limited task specification interpreter. An already available program for symbolic constraint solver has been used to test the output from the skeleton matcher. A parallel development of grasp planner took place and this module in conjunction with the skeleton matcher gives a complete scope for developing constraint propagator.

The skeleton matcher treats the plan steps in terms of its input and its output. At each step, there are restrictions imposed by the execution of this step that are invariant of the specific details. The skeleton matcher expands each plan step to incorporate the propagation of these invariant constraints and leaves other decisions which depend on specific detail to be expanded later by the planning module.

A skeleton is specified by a geometric description of objects and their state in a world and by two sets of constraints: a set of applicability constraints and a set of propagation constraints. The skeleton gets instatiated by finding a match betwe : the geometric description of the object and the known world state.The applicability and the propagation constraints are modeled in terms of variables, which because of the geometric matching get instantiated to specific values relevant for the current world.

In light of these steps, we will begin with discussing the geometric modeler and the task specification before we outline the algorithm and scope of our skeleton matcher.

3.1 Geometric modeller

The modelling system that has been used to represent the objects and the spatial configuration of the objects in the world bears a significant effect on the types of real world operations that can be used to demonstrate the planner.

Primitive objects in the modelling system are constructed by sweeping either a rectangular cross-section or a circular cross- section along a straight line with constant sweeping rule. That is, they are rectangular prisms and circular cylinders. Each part of the primitive that refers to any physical dimension like the length and breadth of rectangular cross-section or radius of circular cross-section or length of the sweep, can be modeled as a nominal value and a deviation value. These elements can be holes. This allows for representation of circular or rectangular recesses in solid objects.

Primitive elements can be affixed to each other by a set of transformations that allow us to define a solid object consisting of several elements. The transformations, however, do not allow for any uncertainty to be incorporated as a part of modelling.

3.2 Task specification

Task specification provides flexibility for the user in specifying tasks. Several attempts at this level have been made of which Lieberman and Wesley [Lieb 77], Lozano-Perez [Lozano 76] and Popplestone, Ambler and Bellos [Pop 80] are significant.

In the current implementation of ATLAS, we have not concentrated on any specific syntax or semantics of task specification. Of the many possible predicates used in specifying any assembly operation, two predicates PLACE and INSERT are most frequently used. They have been implemented with sub-predicates like AGAINST and INTO to specify constraints imposed on the feature elements of the objects in question.

For example, two operations in our example assembly in section 2 will be specified as following:

1. (Place LID BOX (Against (FACE0 LID) (FACE1 BOX)))
2. (Insert BOLT1 (Into (HOLE1 BOX) (HOLE5 LID)))

In the assembly graph representation, each operation has been viewed as a process of introducing some relationship between objects. 'Insert' means an act of introducing a cylindrical interaction. The feature elements that participate in the interaction are specified as arguments of sub-predicate 'Into'. A cylindrical feature of the first argument of 'Insert' BOLT1 gets cylindrical freedom with respect to the cylindrical features of the prisms HOLE1 of object BOX and HOLE5 Of object LID. Other interactions may also get introduced as a by-product of this action. We will talk about them in skeleton matching.

'Place' in a similar sense introduces interactions of types indexed according to the type of features specified in the arguments of the sub-predicate. For example, FACE0 and FACE1 for the sub-predicate 'Against' in the above example index into planar interaction.

An extension on this line will yield a task specifier which can coherently map the specification into relationships of the right kind in the assembly graph.

3.3 Skeleton matching

The process of skeleton matching can be divided into the following sets of sequential operations:

1. Inference of goal position of the 'primary object', usually with respect to an already existing object in the workspace. The first object to be introduced in the workspace has its position variables associated with variable names which will get instantiated to specific values by the layout iterator part of skeleton matcher.
2. Identification of applicability constraints. At this stage some tasks that will not succeed under any circumstance are identified. However, generation of applicability constraints is postponed until the end.
3. Finding out all 'interactions' that the current object acquires. In each task specification, there is a primary interaction inferred from the syntax and arguments and there are other interactions that get introduced as side effects of the primary operation. This portion updates the relationships for the primary interaction as well as other interactions in the 'assembly graph'. The assembly graph stores particular information about each interaction in terms of the description of prisms and features participating in the relationships.

   (steps 1,2 and 3 truly belong to user interface module.)
4. Identification of primary skeletons for this operation. At present, the library of skeletons include:

i) Gross Motion
ii) Grasp
iii) Bolt in Hole
iv) Ungrasp
v) Vertical align without sensing (i.e. stacking)
vi) Catch-all synthesised fine motion

In the context of each of these skeletons, grasp is introduced with creation of variables representing the point of grasp for the object. The propagation constraints are introduced tagged with any fine motion skeleton like (iii), (v) and (vi). It is at these steps that propagation constraints acquire physical meaning. All other skeletons have nothing associated with them at this stage.

5. Generation of applicability constraints and propagation constraints which are posted with the corresponding fine motion skeleton. The 'assembly graph' is scanned with specific time-stamp to find the uncertainties associated with each object in question. They help to establish the object's actual position with respect to another object in the 'assembly graph'. Position evaluation of objects with respect to the stationary frame is required because some part of the uncertainty has been modeled with respect to the position of the object in stationary frame.
6. Updating the global skeleton graph which keeps track of the planning islands introduced so far and the objects associated with each of these islands. If any constraints are imposed on them, they are also available in a form acceptable by the constraint solver.
7. Updating the global logical-time variable to show completion of a set of operations. Each skeleton is associated with one logical time unit. Hence normally, one object movement operation gets incremented by a unit of five corresponding to (gross-to-grasp, grasp,gross-to-goal,fine-motion,ungrasp).

In the present structure of the skeleton matcher, this module also acts as the executive module, accepting the task specification and generating a preliminary plan for the task. The constraint propagator will be invoked by this module which in turn maintains a generator/ iterator relationship with other modules. Thus constraint propagator will return to skeleton matcher only if backtracking has caused it to pop up to the level where a new skeleton is required. Usually at this stage introduction of sensing operations at a point where backtracking identified a variable causing the failure, may solve the problem. However, if this fails then a new layout is generated for fresh iteration.

Thus, the skeleton matcher will iterate over a set of feasible layouts until success or exhaustion of all possible layouts and failing to come up with a satisfactory plan.

4. Links among planning islands

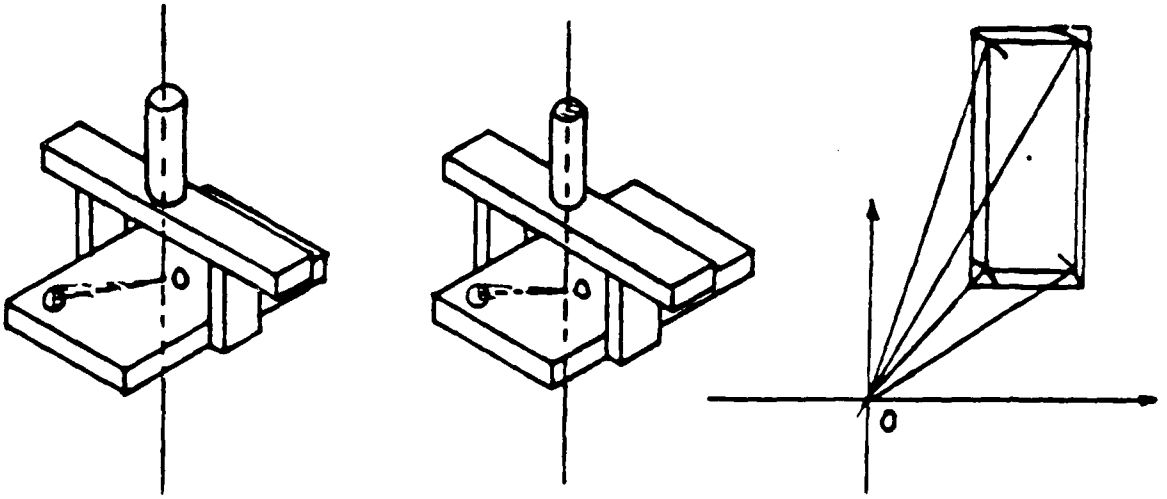
The ATLAS approach to task planning is to first model all plan steps at a broad level (as explained in last section at skeleton level) and generate all constraints which are invariant of successive refinement processes. The effects of constraints are propagated throughout the complete plan. Each step is then refined into further details. Again constraints are propagated and in case of failure, dependency-directed backtracking is invoked.

Isolation of dependency links is a major source of imparting knowledge to the planner about means of achieving a task. A planner which carries out plan generation for each module in isolation obviously has less chance to succeed than one that decides things in consultation with other modules. Usually, each module will have specifics to decide what can be divided into intra and inter-module effect. For example,in the grasp module, exact path of movement of the manipulator from approach-of-grasp-point to the grasp point is an intra-module decision as long as the module propagates the guard volume to carry the operation, but choice of actual grasp point is a inter-module decision.

In addition to isolating the dependency links for grasping and gross motion modules, we will present ways to model them using manipulator error characteristics presented in section 2.3.

4.1 Grasping

Decisions in the grasping module are constrained by the location and environment of the part at the initial position and at the final position. These constraints are intra-module decisions if the grasping module specifies the guard volume and the planner can give information to the grasp planner about orientation of the part at the initial and final positions. But these restrictions by themselves do not model the grasping module's effect on the environment.



a.Effect of Grasping point b.Approx. of rotn. unc.

For a planar interaction

fig 5

Consider the task in the assembly example in section 2.2. The choice of grasp position for the LID in operation 9 affects the success of the Bolt-In-Hole task in operation 18. The applicability constraint associated with operation 18 specifies that HOLE5 of LID and HOLE1 of BOX have to be aligned to sufficient extent so that the operation of Bolt- -in-hole can be successful. If the wrist of the manipulator can be located within ± 0.2 degrees then depending on the position of the grasp which determines the arm, HOLE5's error ball will be different.

If we recall the error characteristic model of the manipulator, the first order terms of the uncertainty are:

$$
\begin{aligned}
Position = & \Delta[T_{link2,world}] * .. * [T^{nom}_{link'i','link'i-1'}] * .. * [T^{nom}_{grip-obj,link'n'}] \\
& + ... \\
& + [T^{nom}_{link2,world}] * .. * \Delta[T_{link'i','link'i-1'}] * .. * [T^{nom}_{grip-obj,link'n'}] \\
& + ... \\
& + [T^{nom}_{link2,world}] * .. * [T^{nom}_{link'i','link'i-1'}] * .. * \Delta[T_{grip-obj,link'n'}] \\
& + \textit{higher order terms.}
\end{aligned}
$$

Of these terms, the last of the first order terms involving transformation from 'grip-obj' to link'n' will contain solely rotational parts. The manipulator error characteristic is modeled such that the uncertainty due to the rest of the terms contributes to translational uncertainty of the object's position and this term contributes to the rotational uncertainty of the object. This allows grasp points to become dependent on the task's applicability constraint. The uncertainty expressions are simplified.

Recall that the uncertainty of an object is the intersection of the space spanned by residual degrees of freedom and the error ball of the manipulator. The rotational part of the uncertainty can be resolved along the residual freedom variable space. Fig 5 contains illustration for an object which gets planar residual freedom. Note that uncertainty due to the rotational term has been approximated by a rectangle enclosing the translational uncertainty with average rotational error for the point of interest in the given object.

4.2 Gross motion

Decisions in the gross motion module belong to the intra-module category as long as there exists a path and propagation of proper guard volume for the start and the goal position has been incorporated.

Usually, the goal position of a manipulator when solved using inverse kinematics, results in multiple solutions. The choice among them can be guided by considerations for uncertainty at the final position.

Considering the uncertainty expression from 2.3, we realize that the uncertainty at the goal position is a function of the nominal values of link orientations for the goal position. The planner should choose a configuration that involves minimum contribution from most of the free variables of the manipulator. Finding a configuration that contributes least to the uncertainty involves examining effects of each of the first order uncertainty terms in relation to the residual freedom variables. Therefore, transformations that convert either the residual freedom space to the joint space or vice-versa are required. Such transformations can be obtained in two ways. First, by following the assembly graph to the stationary frame and then the manipulator chain to the joint space. Or second, recognising that the manipulator's assembly graph and the 'assembly graph' at the moment of ungrasp operation will have the object as a common node. Using the latter method, number of transformation multiplications are reduced considerably. With this method, the choice among multiple positions of the manipulator at the goal position can be limited.

Appendix J

PROGRAM FOR SIMULATION OF TWO-LINK MANIPULATOR WITH FLEXIBLE TENDONS

This appendix describes the derivation of the equations of motion and the essential programs and subprograms used to simulate the Stanford two-link arm with flexible tendon drive. The codes themselves are available on request as a separate pamphlet.

Program LINKM: Linkm is a simple test program which calls a linearized and two non-linearized equations of motion subroutines and calculates the system states, linearized F, G and H matrices (used in control system design as a first order, linear system model: xdot = F x + G u, y = H x), MIMO transfer function poles and zeros, and eigenvalues and eigenvectors of the linearised plant.

Subroutine NONLINM: Nonlinm contains the full nonlinear equations of motion of the two-link arm with flexible tendons and viscous damping in the drive motors. The equations of motion were derived automatically by the symbolic manipulation program MACSYMA and automatically translated into FORTRAN code.

Subroutine NONLIN: Nonlin contains the full nonlinear equations of motion of the two-link arm with flexible tendons and viscous damping in the drive motors. The equations of motion were derived by hand and are believed to contain errors that the MACSYMA derived equations do not have.

Subroutine LINEARM: Linearm contains the linearized equations of motion of the two-link arm with flexible tendons and viscous damping in the drive motors. The linearized equations were derived automatically using MACSYMA.

A complete printout of these programs is available on request as a separate pamphlet.

Appendix K

ELECTRONIC INTERFACE MODULE FOR A PNEUMATIC "PICK AND PLACE" GRIPPER

Abstract

The "One link flexible manipulator" developed at the Department of Aeronautics & Astronautics accomplished the first "pick and place" maneuver by using a solenoid operated mechanical gripper (Schmitz 1985).

Being the "first cut gripper" it was able to "pick and place" a ping-pong ball. To enable the "pick and place" maneuver with a heavier mass, a pneumatic gripper was designed.

A separate experiment was performed to measure the response time of the pneumatic gripper when a pressure is applied, and the recovery time of the gripper when the pressure is removed. Based on the results of this preliminary experiment a gripper operating electronic interface module was developed that allows the "one link flexible manipulator" to perform the fastest "pick and place" maneuver for varying mass.

One of the possible future uses of this gripper will be in the field of the adaptive control.

Introduction

The "pick and place" maneuver as performed by the "one link flexible manipulator" with the pneumatic gripper could be divided in several sequences:

1. The manipulator orientation towards the object to be picked up (in our case the disc).
2. Gripper actuated and pushed down into the disc's hole.
3. Pressure applied to the inflatable rubber finger.
4. Gripper actuated and pulled up carrying the lifted disc.
5. The manipulator orientation towards the "placing" place.
6. Gripper actuated and pushed down.
7. Pressure released from the rubber finger.
8. Gripper actuated and pulled up ready to move to pick up a "new" object (starting once again at sequence #1)

The time needed for sequences #2 and #6 is listed by the pneumatic valve manufacturers and is considerably short (approx. 5-10 msec.)

This movement is down oriented, therefore is short with or without the disc.

We performed a preliminary experiment to determine the minimum time interval needed between sequences #3 and #4 and sequences #7 and #8 as well.

According to the results of this preliminary experiment (as described in section 2) an electronic command interface module was developed and connected between the computer (LSI 11/23) and the gripper itself.

To determine the time interval needed for the rubber finger "to grip", the following measurement was done:

Having the rubber finger inside the disk, an electric command opened the pneumatic valve inflating the rubber gripper. The force F_1 applied by a solenoid actuator pressed the disk against the vertical steel wall, keeping it horisontal due to the friction between the disc and wall. The force F_1 -released a ΔT_1, time interval after the pressure was applied to the rubber finger. This ΔT_1, was adjustable.

If ΔT_1 was long enough the disk remained attached to the gripper.

If ΔT_1 was too short the disc fell down before the gripper finger gripped it.

Finally, using a 14 mm diameter rubber finger and an 18 mm diameter hole inside the disc are measured a minimum 125 msec time interval needed for a proper operation.

By using a 16 mm diameter rubber finger this time interval ΔT_1 decreased to 5 msec. The amount of time ΔT_2 needed to release the pressure from the rubber finger (through an attached and electrically operated exhaust valve.)

A two channel memory osciloscope was connected as follows:

Channel 1 connected and triggered by the electrical step command applied to the exhaust valve to release the pressure from the gripper.

Channel 2 connected to the horisontal base of the experimental set up.

The disc was connected (when hanging free on the gripper) to + 5V.

Once the electrical step command applied, the pressure was released and channel 1 triggered. The disc felt down and by touching the base triggered channel 2

2. The preliminary experiment

As mentioned above the goal of this preliminary experiment was to measure two time intervals. both time intervals were crucial for a fast "pick and place" maneuver performance.

First, it is important to know the amount of time needed for the rubber finger "to grip " the disc firmly after the pressure is applied.

This time is the minimum necessary to keep the arm steady above the disc after having the gripper actuated (pushed down) and before pulling-up pneumatically the "gripper and disc system".

Also important is to know the time interval needed to release the pressure from the rubber finger to allow a smooth exit of the finger from inside the disc before it is moving to the new "pick up" address.

$$\Delta T_4 = \Delta T_2 + \Delta T_3$$

Where

ΔT_2 = the amount of time needed to release the disc after the pressure release

ΔT_3 = the free falling time of the disc.

Finally we got that $\Delta T \approx 35$ msec.

These results were used to design a fast command interface module to provide the computer control for the gripper.

3. The interface circuit

The circuit drawing is shown in Appendix C. The circuit uses two output ($D_0 and D_1$) from the MINC computer. the software that operates the experimental gripper is called "EPULSE". D_0 connects to pin A of the circuit and D_1 connects to pin 7.

An initialising network that consists on an R.S.F.F. network (IC-D - 74132) drives the two valves, A and B. Valve A pushes down the gripper and valve B lifts it. For that reason they are operating in a 180° phase shift.

Through pin A, a pulse with an adjustable width is provided by the computer. The pulse width will determine the delay between the gripper vertical movement and the pressure connection or release to or from the rubber gripper finger. Being an ON-OFF type procedure, the pressure connection and release is controlled by a JKFF (IC-C).

This JKFF is clocked by the command pulse coming through pin A. The gripper vertical movement is controlled by the computer command comming through pin A and pin F.

4. Conclusions

The gripper has not been mounted yet. However, it performed a functional test operation successfully. Based on this test operation, the gripper qualified to be used on the "One link flexible manipulator" for a "pick and place" maneuver.

Appendix L

DIRECT VELOCITY MEASUREMENT USING AN OPTICAL ENCODER

Abstract

To achieve better control of a multilink robot arm it is necessary to measure both angular position and rate of each joint.

The easiest way to measure the position is to use a sensor mounted on the joint. One of the more accurate techniques to measure angular position involves the optical shaft encoder.

Usually, the rate is measured by using a tachometer mounted on the joint or an inertial type rate sensor that could be mounted on the link itself. In the first case, the tachometer measures the link's rate related to its' joint. A rate sensor mounted on the link itself measures the absolute rate of the link related to the base.

In both cases the extra rate sensor adds extra weight, extra moment of inertia, and extra cost to the system.

In order to avoid that, we developed an interface circuit that uses the existing incremental optical shaft encoder's output to provide a rate measurement.

The technique that was used is a quantization of the measurement angle updating the data at a given time interval. This time interval was selected to provide the required accuracy. Reciprocal rate is found without differentiating by measuring the time for a given displacement. Velocity is then computed using a single division operation in the control computer.

Special thanks to Mr. Lawrence Pfeffer for his collaboration in developing a very versatile and accurate motion measurement card based on a position control circuit (developed by Mr. Pfeffer) and on this velocity measuring circuit described herein.

At this time Stanford University is checking the possibility to patent this circuit.

1. Introduction

The interface circuit reported here was developed as part of an ongoing research related to the "Two Link Flexible Manipulator" conducted by Professor Robert H. Cannon and Mr. Michael Hollars, Ph.D. candidate.

Since the angular (rate) velocity is the derivative of an angle, the rate is often a noisy signal if differentiated from an angular measurement. Using the angle quantization method we can eliminate this problem.

The optical shaft encoder consists of a circular glass disc imprinted on the external perimeter with equally divided imprinted rows of broken concentric arcs. A light source (usually a LED) is assigned to each row with a detector on the opposite side of the glass disc. The disc is attached to the rotating shaft. The light detector can be exposed to the light coming from the corresponding LED or covered by the rotating disc, causing a series of electric pulses. By counting these pulses we can determine the shaft position. Being a non-contact multiturn digital device, the optical shaft encoder does not influence the motion but provides a very convenient output.

We considered two main possibilities to measure the rate by using the optical shaft encoder.

The first was to measure how many encoder count pulses occur in a selected time interval (time quantisation).

The second was to measure the amount of time in an angular interval defined by two consecutive encoder pulses (angle quantisation).

In time quantisation, the time interval must be long enough to accumulate a meaningful number of angle pulses; the one count uncertainty, due to asynchronous operation must not represent too large a rate error. This limit is achievable only when either the rate or the encoders resolution (counts per revolution) is high.

In angle quantisation an accurate angular interval is given by one complete cycle of a single encoder channel. (This works for 1 or 2 channel encoders equally well). We then measure time by counting pulses based on a stable crystal oscillator. We can then ensure that the one count uncertainty is small by using a high frequency clock with a programmable clock divider circuit. This gives us the flexibility to work in many ranges of speed and resolution.

For this method, the error induced by the one count uncertainty is more significant at high angular rates. In this instance, too few clock pulses will be accumulated in the given angle. Maximum rate is usually well known, however, and the clock can be programmed to a frequency such that the maximum error is still within acceptable limits.

The only cost is the requirement for a single digital division operation, (which could be done as a look-up table in a ROM). The key benefit is that this reciprocal method yields greatest resolution at low speeds, where it is most useful.

Furthermore, the control loop requires a higher accuracy in the rate measurement at low rates, rather then at high rates. By using a 15 bit counter in our interface circuit and a programmable clock oscillator, we succeeded in meeting the requirements at high and low rates.

2. The Technical Specifications

According to the "Two Link Arm" system design, the maximum expected rate was $600°/sec$ and the minimum rate $.05°/sec$. The interface circuit described here was developed according to these specifications and according to the encoder's specification primarily imposed by the position measurement requirements.

The minimum and maximum limits of the rate measurement could be easily shifted in both directions by adjusting the clock frequency used inside the interface circuit.

This circuit is designed to interface with two most popular shaft encoder outputs: square wave output (TTL level) or line driver output. It is also very easy to modify the circuit to interface with any other encoder outputs.

The output is a tri-state output connected to a digital I/O card for the computer (PDP 11/24).

3. The Velocity Measuring Circuit

3.1. General

Having this velocity measuring circuit developed we decided to design a complete motion measurement card included also a position measuring circuit (developed by Mr. Lawrence Pfeffer).

This card circuit drawing is shown in Appendix A.

3.2. Circuit Block Diagram

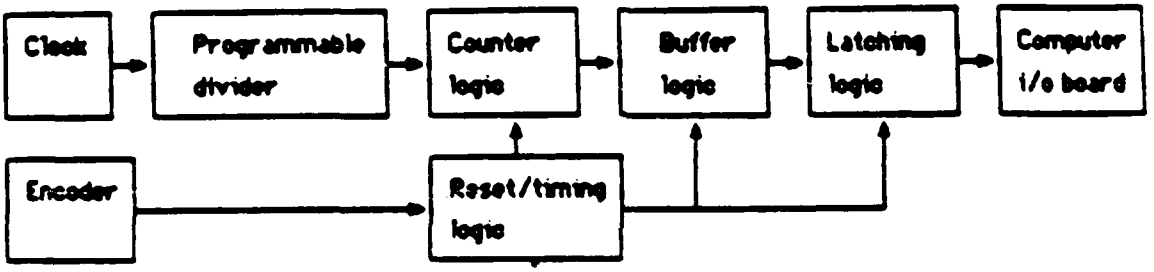


Figure 1. Interface circuit block diagram.

3.3. Circuit Description

3.3.1. The Clock and the Programmable Divider

These two blocks are included in a SARONIX 8640 chip (IC-7A). The logic configuration at its control pins will determine the divided frequency to be used for time measurement in an angle interval.

3.3.2. The Encoder Pulses

The encoder's output is buffered either by a line receiver (in our case I.C.9AIC-9A DS 26LS32 for our Teledyne Gurley encoders) or a Schmitt inverter (IC-9B-74LS14) triggered by the clock's basic undivided frequency latches the encoder output triggering the timing logic and reset logic as well (IC-6C, IC-7C, IC-7D, IC-5C)

3.3.3. The Reset/Timing Logic

The first one shot of IC-6 is triggered at pin 2 by the latched encoder pulse's up-going edge. Its Q output (pin 4) closes temporarily the ANDgate (IC-7D pin 2) blocking the clock pulses coming from IC-7A pin 7 for approximately 50 nsec. Meanwhile, the down-going pulse at pin 4 of IC-6C triggers through pin 12 of IC-7D the first one shot of IC-5C. Its output clocks after approximately 50 nsec (at the up-going edge of $\overline{Q}$ at pin 12 IC-5C) the IC-2C and IC-4C buffers and latches their outputs (the actual state of the counter logic output, which is stable having the clock pulses blocked previously).

At this time the second one shot of IC-5C is triggered and its output $\overline{Q}$ at pin 4 IC-5C blocks for 50 nsec the unsynchronized "computer ready" signal (coming from the computer) that assures a stable data at the second final latch outputs (IC1C and IC3C) when the computer will read it.

The second one shot of IC-6C is triggered to cause a reset to the counters after the latching procedure is accomplished. These delays described above are short enough not to influence the data and set to operate in the right timing.

3.3.4. The Counter Logic

The counter logic consists of 4 cascade connected asynchronous counters (IC-1D, IC-2D, IC-3D and IC-4D). A carry signal (pin 12-IC-1D) is used to reset through IC-7C the counters if they overflow.

Actually, the counters overflow if the velocity is under the lower selected limit. This limit is always the lower detectable velocity required by the control system. Below this limit the velocity is considered zero.

Only 15 bits of the counter logic are used as data for the velocity magnitude. One input to the buffer logic and latching logic is used to provide the sign information, coming from pin 8 of IC-5A.

The encoder has 2 channels with 90° phase shift between them (a quadrature output). The sign logic identifies if channel A leads channel B or channel B leads channel A. In one case we are dealing with a positive velocity, in the other case with a negative one.

3.3.5. The Buffer Logic

As mentioned in 3.3.4, IC-2C and IC-4C are bit buffer/latch networks (74LS374). They are buffering the counter logic from the latching logic providing stable data during the latching to the computer I/O card.

3.3.6. The Latching Logic

The latching logic (IC-1C and IC-3C) includes two 74LS374 chip. It latches the stable updated data to the computer I/O card.

3.3.7. The Position Measuring Circuit

The position circuit is almost totally separated from the velocity measuring circuit, except that they are showing the encoder outputs the clock chip and the same bus to the computer I/O board sharing.